6433 PROGRAMMABLE SIGNAL PROCESSOR

USERS GUIDE

Date : Feb. 84

Issue: F
JSH

Part No: HA 075417 U003

Turnbull Control Systems Limited
Broadwater Trading Estate Worthing Sussex BN14 8NW
Telephone Worthing (0903)205277
Telex 87437

6433 USER GUIDE ADDENDUM SHEET

Software issue 1/1E

Remove FS and SETFS operators; forward space in Line Edit mode is now achieved by rewriting characters.

Block framing characters STX and ETB added to Save utility.

Software issue 1/1F

XON-XOFF added to Save utility.

Change to ULIST, method of paging.

Software issue 2; release 1

The user program memory has been expanded to 8K. This memory is situated on the ROM/RAM emulator daughter card (assembly AC 075114). This card is fitted with one 8K EEPROM chip and one 8K RAM chip. No difference should be seen by the programmer, except that the FORTH prompt will show an additional 4096 bytes of free memory.

About this Manual

This manual describes the programmable characteristics of the 6433 signal processor.

The text assumes that the reader has a good working knowledge of the 6432 signal processor, upon which the 6433 is based. Readers without this knowledge are strongly advised to refer to the 6432 technical manual before attempting to proceed with the 6433.

The 6433 is programmed in a version of the "FORTH"* language, developed by TCS specifically for use with the 6433. TCS FORTH differs from other standard FORTH implementations in that:

a) it is considerably simpler, not requiring any of the features found in other implementations (e.g. it uses no disk file I/O)

b) it handles numeric data exclusively in floating point format, thus removing the need for the programmer to consider binary point positioning.

The manual describes the TCS FORTH implementation, and provides examples relating to the use of the language. An appendix shows a documented application program for a typical process control problem.

Readers who already have programming experience (not necessarily in FORTH) should find the manual self-sufficient. Readers who have no previous software knowledge will probably find it necessary to refer to a standard introductory text on FORTH, which will provide an introduction to programming and the concepts of FORTH as a language. Readers who refer to a standard FORTH textbook are advised to use the standard text only to obtain a basic understanding of the concepts prior to proceeding to this manual.

The manual includes a section in the appendices entitled "First steps in using a 6433", which is a step by step description of setting-up, powering-up and preparing to program a 6433. This section is intended to be used once readers have a general understanding of the 6433 programming language, and are ready to commence practical exercises.

Related Literature

a) 6432 Technical Manual:
Readers will find it necessary to have this manual available for ready reference.

b) System 6000 Communications Handbook:
This handbook is required only if the user intends to communicate with the 6433 via the RS422 serial port.

c) 6433 Programming Terminal Users Guide:
This manual is required only when using the TCS intelligent programming terminal.

d) Introductory Text Books on FORTH:
'Starting Forth' by Leo Brodie - Prentice Hall

* FORTH is a registered trademark of FORTH INC.

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

APPENDICES

Section 1 GENERAL DESCRIPTION

1.1 Introduction

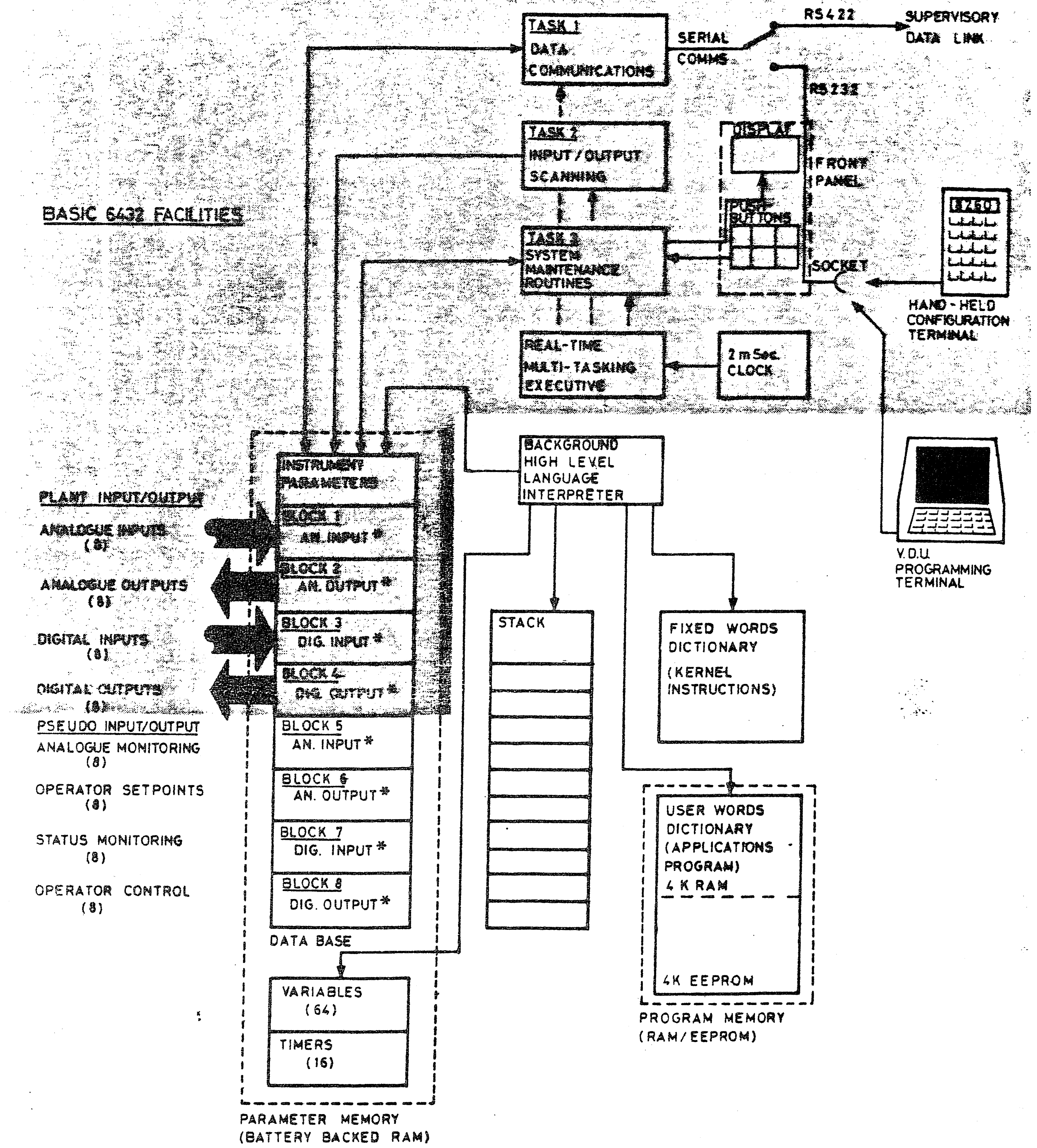
The 6433 Programmable Signal Processor is based on the hardware of the 32 channel 6432 Signal Processor. The latter is primarily intended for analogue and digital data acquisition and display but includes the ability to set alarms on selected input and link these to specified digital outputs.

The 6433 features a software enhancement which allows both arithmetic and boolean computation by means of a stack-oriented interpretive programming language based on a subset of FORTH. This includes various enhancements to provide floating point arithmetic and 16 internal timers. The language is highly compact, and statements are entered in a "Reverse-Polish" mode.

The unit is intended for arithmetic computation, special control functions, logical manipulation and simple ramp generation and sequencing applications. Timing functions are carried out to a precision of 2 milliseconds under control of a real time clock.

Input/output capability is the same as for the 6432 (i.e. 4 blocks of 8 channels where blocks may be analogue or digital inputs or outputs) but in addition there are 4 blocks of 8 pseudo channels which may be used for internal derived values. These are accessible from the front panel and via the handheld programming terminal socket or serial link in the same way as the real input/output channels, and would typically be used to display the results of calculations, status, and to set internal constants.

User memory is 4K RAM and 4K EEPROM. Programs are developed in RAM by using an RS232 "Teletype" compatible device such as a VDU plugged into the front panel programming socket. A program developed in RAM can be "fixed" into ROM from the programming device. The program is, however, always run from RAM and this allows programs to be loaded into RAM, edited and then debugged before the original program is replaced.



SOFTWARE ENHANCEMENTS WITHIN 6433

*NOTE: ANALOGUE AND DIGITAL INPUT/OUTPUT CARDS
ARE SHOWN IN A TYPICAL CONFIGURATION;
ANY COMBINATION MAY BE SPECIFIED.

6433 PROGRAMMABLE SIGNAL PROCESSOR
FUNCTIONAL OVERVIEW
FIGURE 1.1

1.2 Functional Overview

The heart of the 6433 is the database as shown in the functional overview of Fig 1.1. This holds the instrument parameters and channel parameters for the 4 blocks of 8 real input/output channels as for the 6432. In addition it holds the parameters for the 4 blocks of 8 internal or pseudo channels.

This database is scanned continuously and updated with the values of real inputs or computed values; output blocks are updated with new values if a change has taken place to the relevant values in the database. All values are available for display on the front panel under the control of the pushbuttons.

As in the 6432 instrument values may also be accessed via the data communications task. In normal on-line operation this services the RS422 serial link. However plugging in a programming terminal disables the RS422 link and transfers control to the RS232 front panel socket which may be used for either the handheld terminal or a teletype compatible console such as a VDU.

Also held in the database are current values of the 64 variables and 16 timers. Variables are stored in 32 bit floating point format. Timer values are stored as 32 bit numbers with 1 bit corresponding to 2 milliseconds, giving a range of ±4,294,960 sec or about 7 weeks.

User programs are written in a high level interpretive language based on a version of FORTH. Program statements are called "Words" and these loosely correspond to subroutines. A kernel set of the most common functions are resident in ROM in the "Fixed Word Dictionary".

These comprise the common arithmetic and boolean operators, input output routines etc. The user then builds up a program by creating a hierarchy of user words in which both user-created and fixed dictionary words may be nested and strung together. The user words are added to the "User Word Dictionary".

Program words operate on one or more values on a data "stack". New values are put onto the top of the stack and push down old values. They are picked off on a first-in last-out basis so that program statements operate in a "Reverse Polish" format.

The front panel RS232 socket supports two terminal modes. The first is Command Mode which provides the normal access to configuration parameters via 2 character mnemonics. The second is Programming Mode which provides access to the FORTH editor for entering and modifying programs. This mode is protected by a security number associated with a user name.

1.3 Programming Considerations

A stack-oriented language imposes a structured approach to applications programming. Program statements or "Words" can only call other routines that have already been defined and are recognised by the interpreter. This implies "top down" formulation of the problem with "bottom up" implementation of the program.

The first step is to define the database. This will comprise real inputs and outputs according to the hardware configuration, and then internal or "pseudo" inputs and outputs for derived values and constants requiring access from the front panel. It should be noted that derived values for display should be set up as pseudo inputs because this allows the facility of setting high and low alarms as for real inputs. Conversely operator-set constants should be set as outputs so as to allow use of the raise/lower buttons or handheld terminal for changing values.

Intermediate results used in more than one place in the program but not required for display are conveniently set as variables. This has the advantage of reducing dependence on the stack and can allow individual program statements to be more self contained.

The next step is to draw a flow chart for each independent task. All tasks residing in the instrument must then be incorporated in a master loop which performs scheduling; usually it is sufficient to cycle sequentially through all the tasks and enclose them in a BEGIN...END structure. Two preliminary tasks are also generally required to set up initial values of constants and to reset timers, counters and flags etc.

The third step is to divide the flow chart into sub-routines of a suitable size to be defined as program words. It may be desirable for clarity to restrict word length to one line on the terminal device, to a length set by the WINDOW operator. Longer statements can be achieved by stringing together several words. Words should as far as possible be self contained.

The final step is to translate the flow chart into code and enter the program, noting that words lowest in the hierarchy must be entered first.

For simple analogue computations a block diagram approach is often more appropriate than a flow chart. An example is shown in the Appendix.

Individual parts of the program may be tested at any stage by executing in immediate mode, simply by typing in the appropriate Word name.

1.4 Instrument Configuration and Terminal Operation

1.4.1 Instrument Set-up

The 6433 internal switches are set up in the same way as for the 6432. Note, however, that an extra switch S1 no. 1 determines whether the baud rate of the front panel RS232 socket is 300 baud, as for the handheld terminal, or as selected on S1 no(s). 2 to 4. A VDU or a teletype which has a data buffer, may use 9600 baud.

1.4.2 Parameter Set-up

Instrument parameters for the 4 real input/output blocks should be set up as for the 6432. A further set of 4 blocks may be defined for internal pseudo channels.

1.4.3 Terminal Configuration

A special lead is required to plug a VDU or teletype into the front panel socket. Connections for the latter, with designation referring to the terminal are as follows:

6433 Designation	HHT Socket	25 way Connector	RS232 Designation
0V(Signal Ground)	H	7	Signal Ground
Hand Held Terminal Sense	E(link to 0V)		
Transmit (from Terminal)	B	2	Trans. Data (from Terminal)
Receive (to Terminal)	F	3	Rec. Data (to Terminal)

It is necessary to ensure that the terminal data format is set up correctly, viz:

1 Start bit
7 Data bits
1 Parity bit (even)
1 Stop bit (2 at 110 Baud)

Use of the Line Editor in Programming Mode requires that a suitable character (normally TAB) is defined for Forward Space.

1.4.4 Terminal Operating Modes

At power up the instrument executes the word MAIN if this is resident in the user memory. A terminal plugged into the front panel allows access to parameters as for the 6432.

It is convenient to use a VDU to emulate a Hand Held Terminal when configuring the 6433 as described in section 2.6.1.

Programming Mode may be entered by keying CTRL P. Access is only granted on acceptance of the valid 3 digit security code outlined in section 2.6.1.

Entries from the programming device are terminated using the RETURN key in the usual way.

The contents of the Fixed and User Word Dictionaries may be examined using the FWORDS and UWORDS commands respectively. Available memory is printed when entering programming mode. Handheld terminal mode is restored by typing CTRL Q. Detailed facilities are described in section 2.6.

A useful diagnostic feature is the ability to examine the top stack entry. This is done by entering a full stop which causes the top of the stack to be printed on the terminal. Note that the top of stack value is lost; if this is undesirable then the DUP command may proceed the full stop.

This feature is commonly used when testing program statements in immediate mode.

Section 2 THE SIGNAL PROCESSING LANGUAGE AND ITS OPERATION

2.1 6433 Software Structure

The basic software structure of the 6433 Programmable Signal Processor is illustrated by the functional overview of Fig. 1.1 where it can be seen to consist of the following main components:-

2.1.1 Executive

The kernel of the 6433 software is a real-time, multi-tasking executive which schedules the 3 system "tasks" on an interrupt driven basis derived from a 2ms real-time clock. As well as controlling this task scheduling scheme the executive is also responsible for multiplexing the front-panel displays.

2.1.2 Task 1 - Communications

The highest priority task in the system is the scheduling of the communications routines. The receive character handling routine is scheduled whenever the UART generates an interrupt after receiving a character via the Hand-Held Terminal or RS422 data link. The transmit character handling routine is scheduled when data has to be sent to the 8260 Hand-Held terminal or RS422 data link and the UART generates an interrupt after each character has been transmitted.

2.1.3 Task 2 - Input/Output

The second highest priority task is scheduled whenever the analogue or digital inputs are to be scanned, or whenever the analogue or digital outputs are to be updated. With analogue inputs, for example, the A to D conversion, Prime Variable ranging, filtering and linearisation are all done within the scheduled Task 2 operation. The 8 analogue or digital channels within each I/O block are controlled by a channel scanning algorithm which works on a basic scan period of 38ms. The rate at which input/output channels are scanned depends upon the types of I/O Blocks fitted, and the number of enabled channels within each Block. During each 38ms scan period, the following occurs on each of the 4 different types of Blocks-:

a) 1 channel is scanned on each analogue input or output Block.

b) Up to 8 channels are scanned on each digital input or output Block.

CHANNELS SCANNED WITHIN I/O BLOCK	38ms SCANNING PERIOD Time in ms →							
	0 to 37	38 to 75	76 to 113	114 to 151	152 to 189	190 to 227	228 to 265	266 to 303
1 (4 enabled)	1	2	3	4	1	2	3	4
2 (3 enabled)	1	2	3	1	2	3	1	2
3 (1 enabled	1	1	1	1	1	1	1	1
4 (8 enabled	1-8	1-8	1-8	1-8	1-8	1-8	1-8	1-8

TABLE 2.1 Example of the channel scanning algorithm in operation

2.1.3 Task 2 - Input/Output /Cont.

This is illustrated by the example given in Table 2.1 where a 6433 is configured as follows:-

(i) Block 1 - analogue input or output with 4 channels enabled.

(ii) Block 2 - analogue input or output with 3 channels enabled.

(iii) Block 3 - analogue input or output with 1 channel enabled.

(iv) Block 4 - digital input or output with 8 channels enabled.

The order in which the channels are scanned in each Block is shown in Table 2.1 where the following points are brought out:-

a) An analogue I/O Block with 4 channels enabled has each channel scanned every 4 x 38 = 152ms.

b) An analogue I/O Block with 1 channel only enabled is scanned every 38ms.

c) A digital I/O Block is scanned every 38ms regardless of the number of channels enabled.

2.1.4 Task 3 - System Maintenance

The lowest priority task is the system maintenance routine which accesses the instrument data base directly using the values obtained by Task 2 and provides the Executive with the current front-panel display data. The maintenance task also scans the front-panel push-buttons to detect operator changes and carries out various other "housekeeping" functions.

2.1.5 Interpreter

Fig 1.1 shows that the interpreter is only activated when all other 3 tasks have been executed and de-scheduled and it can thus be considered as a sort of "background" activity. The function of the interpreter itself is to execute the user defined program sequence, accessing input variables from the instrument data base, processing them as required, and updating the output variables with the results. Input/Output task 2 then continues to access current data from "real world" inputs just as in a 6432, and results of program calculations are available as "real world" outputs in a similar manner.

2.2 Signal Processing Language Structure

The main features of the Signal Processing Language Structure are as follows:-

2.2.1 Language Type

The Language is based on a subset of FORTH, with certain enhancements to allow floating point arithmetic, timing etc. It is interpretive in nature, which means that as each statement is entered it is scanned by the interpreter for syntax errors. An error-free statement will then be compiled before the next statement is entered. This differs from many interpreter-based languages such as Basic, where the statement is recompiled each time it is executed.

2.2.2 Language Notation

Mathematical expressions are entered in Reverse Polish Notation (RPN), rather than in Algebraic format. This gives a number of advantages, viz:-

a) The number of statements required to evaluate a given expression are usually reduced.

b) The need for parenthesis is eliminated.

c) The user is forced to follow the rules of a structured programming language. This improves readability and reliability.

The features of RPN can best be illustrated by means of the following example"-

"Evaluate the expression : A * (B + C)"

This would be done in algebraic format, by working through the expression as it is written from left to right as users of most types of pocket calculator are aware. In Reverse Polish Notation, the expression would be written and evaluated in reverse order, thus:-

C B + A *

This means that the values of C and B are entered first. The + operator always adds the last two numbers entered and leaves a single number result. The value of A is entered next and the * operator multiplies the last two values together, i.e. A and B + C.

2.2.3 Parameter Stack

In order that the RPN structure of Section 2.2.2 can operate, it is necessary to introduce the concept of a Parameter Stack. This is just an array of memory locations, or registers into which the values of parameters are stored and fetched as the interpreter works through an expression. The stack is of the "Push-down" or "last-in, first-out" type, so that operands are always loaded onto the top causing all the existing parameters to be "pushed-down" one location. Conversely, when a parameter is "pulled-off" the top of the stack, all the others move up one location. The example above may now be split up into individual stack operations, thus:-

a) C - causes the value of C to be loaded onto the top of the stack.

b) B - causes C to be pushed down one place and the value of B placed on the top of the stack.

c) + - causes the top two stack entries to be added together and the result placed on top of the stack.

d) A - loads the value of A on top of the stack and pushes down the previous entry, i.e. (C + B).

e) * - causes the top two stack entries to be multiplied together and the result placed on top of the stack, i.e. A * (B + C).

Data is stored in the Parameter stack in the form of two consecutive 16 bit words. These 32 bit data words are used to store numbers in floating point format with an 8 bit exponent and a 23 bit mantissa and a hidden bit. This allows numbers to be held to 9 digit accuracy in the range 10 to the power ±38. However, it should be noted that numbers can only be displayed to 6 digit accuracy.

2.2.4 Statement Entry

Each program statement is a simple one line entry made via the Programming Terminal which is normally a VDU.

Programming mode is entered by keying CTRL P followed by the security code. The desired line length may then be set up using the WINDOW statement. Handheld terminal mode is restored by keying CTRL Q. A VDU may be used to emulate the handheld terminal if desired. Using a teletype instead of the VDU allows printout of a program listing.

2.3 Operands

The parts of a program statement that actually contain the data to be processed or operated on are known as Operands. In contrast to standard FORTH which generally uses integer arithmetic, all 6433 operands are stored as signed 32 bit floating point format. This includes Boolean data and flags, and the interpreter optimises the arithmetic according to data type so that the overload of non-integer operations is only incurred where necessary. The 6433 interpreter supports several different types of operands which are described in the following sections.

2.3.1 Global Variables

Very often, it is required to save the intermediate result of a calculation for later use, but not to take up storage space on the stack. This is achieved by accessing one of 64 dummy operands or general purpose memory registers referred to as variables 1 to 64. Each of these memories is a full 32 bit register and stores data in floating point format. The data in any of the memories can only be accessed from within a program and cannot be displayed on the 6433 front-panel, the 8260 terminal, or via the RS422 data link.

2.3.2 Local Variables

Routines requiring repeated access to derived values can result in either wasteful use of Variables or intricate manipulation of the stack which may be difficult to follow in a line of FORTH code without a stack progress diagram.

This problem is alleviated by allowing the allocation of up to 8 stack values to be used as Arguments within a word definition. At the same time an 8 position results area is reserved at the top of the stack. These results can similarly be used as local variables within the routine. On leaving an Argument-Result structure a number of entries from the Results area can be left at the top of the stack so as to be passed on to the next routine. All other Arguments and Results are dropped, together with any residual stack entries above the Results area. These Arguments may be accessed locally within the word in the same way as global variables.

Argument definitions may be nested, but only these local to the current definition may be accessed.

This use of local variables for Arguments and Results makes the writing and readability of re-entrant code significantly easier.

2.3.3 Timers

When the 6433 is being used in sequencing operations, it is usually necessary to be able to time the interval between the occurrence of 2 events. For this purpose, the interpreter allows the user to access up to 16 timers, T1 to T16, within a program. Timers are clocked every 2ms and the count is stored in a 32 bit register. This means that the maximum count (positive or negative) is 2^{31} /500 =4,294,960 sec or about 7 weeks. The resolution is the greater of 2ms and 1 part in 2^{24} depending on the magnitude of the current value. Timers may be loaded with either positive or negative values. Once a timer is set up or loaded it is automatically decremented every 2ms. The value in seconds in floating point or exponent format of any of the 16 timers may be accessed by the program and tested to see how much time has elapsed.

2.3.4 Real Input/Output Operands

When the value of one of the 32 input/output channels is to be processed it may be used as the operand of a program statement and accessed simply by block and channel number.

Just as for the 6432 instrument all 32 input/output channels can be displayed on the front-panel, on the programming terminal, or accessed via the RS422 serial link.

2.3.5 Pseudo Input/Output Operands

In addition to the 4 blocks of real inputs and outputs there are a further 4 blocks of internal data which may be nominally identified as analogue or digital inputs/outputs. These are referred to as "pseudo channels" and are intended to be used to allow display of derived values and setting of calculation constants and status bits. The 32 pseudo channels can be accessed via the front panel, the programming terminal or via the RS422 serial link in the same way as real input/output.

Related parameters such as alarms on inputs and high and low limits on outputs are effective as for real input/output channels. Note that filtering and linearisation are not effective on analogue inputs.

2.3.6 Display of Derived Values

Derived values from computations may be displayed by defining them as either pseudo inputs or outputs. However, subject to available channel capacity, they are most conveniently set as pseudo-inputs to allow use of the alarm feature as for real inputs.

It is helpful to consider signal direction in the sense that derived values form inputs to the 6433 display. Note that writing from within the program to a real input is prohibited by the interpreter and will result in a Run-time error.

2.3.7 Operator Setpoints

Constants or status bits requiring to be set at the front panel should be set as pseudo outputs. In the case of analogue values these will be subject to the high and low output limits. If necessary changes to these from the front panel may be prohibited using the Pushbutton Disable facility in the ST status word.

2.3.8 Access to Configuration Parameters

As well as having access to real and pseudo inputs and outputs the program may read and manipulate any of the Configuration Parameters which are normally available on the serial link. These are accessed by block and parameter numbers according to the tables shown in Appendix A3.

2.4 Operators

The parts of a program statement that defines what action is going to be taken on an operand is known as an operator. The 6433 interpreter supports several different types of operators which are described in the following sections.

Stack action is indicated as a list of values before and after execution of the word, in the form:

Stack contents before .... Stack contents after

In all references to the stack, numbers to the right are at the top of the stack. Position in the stack is indicated by a numerical suffix showing the order of data entry; for example the notation n1 n2 is read as "n1 is beneath n2", where n is a 32 bit signed number. Although all stack values are stored as 32 bit signed numbers other data types are distinguished as follows:-

Bn	Input/Output Block number
Cn	Input/Output Channel number
Pn	Parameter number
Vn	Variable number
An	Argument number
Rn	Result number
Tn	Timer number
f	Boolean flag : 0=false, non-zero=true
c	7 bit ASCII character

Section 2.7 gives examples of the use of a selection of operators.

2.4.1 Arithmetic Operators

The interpreter allows a number of arithmetic operators to be used on the parameter stack. These operators work on the top one or two entries of the stack and hence perform full 32 bit floating point calculations as detailed below (see also sections 2.4.5 and 2.4.6 for arithmetic operators acting on variables):-

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
+	Leaves the sum n1 + n2. n1 n2 ..... n3	2 6 9	8 9 -
-	Leaves the difference n1 - n2 n1 n2 ..... n3	2 6 9	4 9 -
*	Leaves the product n1.n2 n1 n2 ..... n3	-2 6 9	-12 9 -

2.4.1 Arithmetic Operators/Cont.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
/	Leaves the ratio n1/n2 n1 n2 ..... n3	-2 6 9	-3 9 -
ABS	Leaves the absolute value n1 ..... n2	-2 -6 9	2 -6 9
MAX	Leaves the larger of top two entries n1 n2 ..... n3	2 6 9	6 9 -
MIN	Leaves the smaller of top two entries n1 n2 ..... n3	2 6 9	2 9 -
MINUS	Negates the top entry n1 ..... n2	8	-8
SQR	Squares the top entry n1 ..... n2	2 6 9	4 6 9
SQRT	Square roots the top entry n1 ..... n2	4 6 9	2 6 9
SIN	Forms the Sine of the top entry n1 ..... n2	-1.15 6 9	-9.12764E-1 6 9
COS	Forms the Cosine of the top entry n1 ..... n2	-1.15 6 9	4.08487E-1 6 9
ATAN	Forms Arctangent (in range ± pi/2 radians) of the top entry. n1 ..... n2	-2 6 9	-1.10715 6 9

2.4.1 Arithmetic Operators/Cont.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
ATAN2	Forms Arctangent(in range ±pi radians) of the 2 argument value in the top 2 stack entries, interpreted as the imaginary (n1) and real (n2) parts. It provides the same function as ATAN (n1/n2) but with the ability to distinguish between all 4 quadrants. n1 n2 ..... n3	-2 6 9	1.89255 9
E	Places value of e at top of stack ..... n	2 6 –	2.71828 2 6
EXP	Exponentiation of top entry. n1 ..... n2	2 6 9	7.38906 6 9
LN	Forms Natural Log (to base e) of top entry. n1 ..... n2	2 6 9	0.693147 6 9
PI	Places value of pi at top of stack. ..... n	2 6 –	3.14159 2 6
INT	Replaces the top of stack value with its integer part. n1 ..... n2	2.786 6 9	2 6 9

2.4.2 Boolean or Logical Operators

The interpreter allows boolean or logical operators to be used on the parameter stack. These operators work on the top one or two entries of the stack and use the logic state of the 16 bit word. For all operators except NOT the top of stack number is first rounded to the nearest integer and limited to the range 0 to + 65535. These operators will however, generally be used with the digital input/output parameters.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
AND	Leaves the AND function of the top 2 entries after rounding and limiting. n1 n2 ..... n3	0 1 1	0 1 -
OR	Leaves the OR function of the top two entries after rounding and limiting. n1 n2 ..... n3	0 0 1	0 1 -
XOR	Leaves the XOR function of the top two entries after rounding and limiting. n1 n2 ..... n3	0 0 1	0 1 -
NOT	Replaces a zero top entry by a 1; otherwise by a 0. n1 n2 ..... n3	0 1 1	1 1 1

2.4.3 Comparison Operators

There is often a requirement to compare the value of two stack entries to determine a subsequent course of action. The 6433 interpreter uses the Comparison Operators for this purpose. They result in the top of stack number being either a 0 or a 1 and are usually followed immediately by an IF operator. An example is shown in section 2.7.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
>	Leaves a true flag if n1 is greater than n2; otherwise leaves a false flag. n1 n2 ..... f	2 6 9	1 9 –
<	Leaves a true flag if n1 is less than n2; otherwise leaves a false flag. n1 n2 ..... f	4 7 3	0 3 –
=	Leaves a true flag if n1 is equal to n2; otherwise leaves a false flag. n1 n2 ..... f	5 5 7	1 7 –
0>	Leaves a true flag if n is greater than 0; otherwise leaves a false flag. n ..... f	8 4 –	1 4 –
0<	Leaves a true flag if n is less than 0; otherwise leaves a false flag. n ..... f	-7 5 –	1 5 –

2.4.4 Signal Input/Output Operators

Most programs at some time require either to fetch data from some source and put it on top of the parameter stack, or to take the data already on top of the stack and transfer it to some destination. The 6433 interpreter incorporates special memory operators to carry out these tasks. Examples are shown in section 2.7.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
GETAN	Moves Analogue data from the input or output Bn1 Cn2 to the top of the stack. n1 n2 .....n3	2 1 –	DATA – –
SETAN	Moves Analogue data from the first stack entry to the input or output. n1 Bn2 Cn3	2 2 DATA	– – –
GETDIG	Moves data from the input or output Bn1 Cn2 to the top of the stack. n1 n2 ..... n3	2 3 –	DATA – –
SETDIG	Moves data from the first entry to the input or output Bn2 Cn3. n1 n2 n3	2 4 DATA	– – –
GETPAR	Moves data from block Bn1 parameter Pn2 to the top of the stack. n1 n2 ..... n3	56 2 –	DATA – –
SETPAR	Moves data from the first stack entry to block Bn2 parameter Pn3. n1 n2 n3	56 3 DATA	– – –

2.4.4 Signal Input/Output Operators/Cont.

Note that when setting Outputs from within a program it may be desirable to inhibit possible interruption from the front panel by using the pushbutton disable feature available in the ST status word. An alternative method, which also prevents changes via the RS422 serial link, is to use the output limits, in the case of Analogue Outputs, or the output masking bits, in the case of Digital Outputs to clamp the data at the desired value. This requires writing to the HO and LO or DS parameters respectively instead of using SETAN and SETDIG. When writing to analogue output limit parameters HO and LO, care is needed to ensure that the High and Low range values HR and LR may not be exceeded, otherwise a run-time error may result. This is done simply by reading these parameters and using the MAX and MIN operators as appropriate.

2.4.5 Global Variable Operators

Internal variables 1 to 64 may be accessed in a similar way to input/output, as follows:

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
GETVAR	Moves data from variable Vn1 to the top of the stack. n1 ..... n2	3 6 2	DATA 6 2
SETVAR	Moves data n1 from the top of the stack to variable Vn2. n1 n2	3 DATA 6	6 - -

Two further operators facilitate implementation of difference equations by forming the difference and the sum respectively between the top of stack value (usually a new analogue input) and a value stored in a variable.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
DIFVAR	Leaves the difference n1-Vn2 between the first entry n1 and the value Vn2 of the variable defined by the second entry. The first entry is stored as the new variable value. Used for numerical differentiation n1 n2 ..... n3	2 1 - V2=5679	-5678 - - V2=1
SUMVAR	Leaves the sum n1+Vn2 of the first entry and the value Vn2 of the variable defined by the second entry. The sum is stored as the new variable value. Used for numerical integration. n1 n2 ..... n3	2 1 - V2=5678	5679 - - V2=5679

2.4.6 Local Variable Operators

The 8 local variables (see section 2.3.2) available within a FORTH word following an ARG definition may be accessed in the same way as global variables. An example is shown in section 2.7.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
GETARG	Moves data from argument An1 to the top of the stack. n1 ..... n2	4 1 R8 5 : R1 2 A5 5 A4 8 A3 7 : :	8 1 A8 5 : R1 2 A5 5 A4 8 A3 7 : :
SETARG	Moves data from the top of the stack to argument An2. n1 n2	3 1 R8 5 R7 -1 : R1 5 A4 8 A3 7 :	R8 5 R7 -1 : R1 5 A4 8 A3 1 A2 - A1 - :
GETRES	Moves data from result Rn1 to the top of the stack. n1 ..... n2	3 4 R8 85 R7 -1 R6 6 R5 11 R4 99 R3 54 R2 8 R1 7 A5 15 A4 25	54 4 R8 85 R7 -1 R6 6 R5 11 R4 99 R3 54 R2 8 R1 7 A5 15 A4 25

2.4.6 Local Variable Operators/Cont.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
SETRES	Move data from the first stack entry n1 to result Rn2. n1 n2	3 1 R8 85 R7 -1 R6 6 R5 11 R4 99 R3 54 R2 8 R1 7 A5 15 A4 25	R8 85 R7 -1 R6 6 R5 11 R4 99 R3 1 R2 8 R1 7 A5 15 A4 25 A3 - A2 -

2.4.7 Timer Operators

Timers 1 to 16 may be loaded with a value or read in a similar way to variables.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
GETTIM	Moves data n1 (seconds) from timer Tn to the top of the stack. n1 ..... n2	3 6 2	DATA 6 2
SETTIM	Moves data n1 (seconds) from the first stack entry to timer n2. n1 n2	3 DATA 6	6 - -
+TIM	Adds data n1 from the first stack entry to the value in timer n2. n1 n2	3 DATA 6	6 - -

2.4.8 Stack Manipulation Operators

With Reverse Polish Notation many calculations can be shortened if the user is allowed to manipulate the entries on the stack directly before using the arithmetic operators. To facilitate this the 6433 Interpreter supports the following stack manipulation operators.

Use of the Argument and Result Operators is introduced in section 2.3.2 on local variables.

OPERATOR	DESCRIPTION	EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
DROP	Discards the top entry n	1 2 3	2 3 -
DUP	Duplicates the top stack entry n ..... n n	1 2 3 -	1 1 2 3
OVER	Copies the second entry over the top entry n1 n2 ..... n1 n2 n1	1 2 3 -	2 1 2 3
ROT	Rotates the top three entries n1 n2 n3 ..... n2 n3 n1	1 2 3 4	3 1 2 4
SWAP	Swaps the top two entries n1 n2 ..... n2 n1	1 2 3	2 1 3
ARG	Defines the number of stack entries (up to 8) to be used locally as Arguments within the current operation. Reserves an 8 position results area at the top of the stack. Normally used at the beginning of the word definition. n ..... A1 A2 (up to A8) R1 R2 to R8	3 9 8 7 6 5 - - - - - - -	R8 - R7 - R6 - R5 - R4 - R3 - R2 - R1 - A3 9 A2 8 A1 7 6 5

2.4.8 Stack Manipulation Operators/Cont.

OPERATOR	DESCRIPTION		EXAMPLE OF STACK BEFORE OPERATION	EXAMPLE OF STACK AFTER OPERATION
RES	Defines the number of parameters (up to 8) to be left on the stack as results of the present operation. Drops all Arguments and any residual stack entries above R8. Normally used at the end of the word definition. A1 A2 (up to A8) R1 R2 to R8 n .....R1 R2 (up to R8)		2	5
			9	4
		R8	8	0
		:		-
		R3	6	-
		R2	5	-
		R1	4	-
		A3	3	-
		A2	2	-
		A1	1	-
			0	-

2.4.9 Terminal Input/Output Operators

The following print control operators permit the transmitting and receiving of data via the front panel RS232 socket. This facility would normally be used during program development.

a) Character Input

KEY Receives a character from the programming terminal and pushes it to the top of stack (0 to 127) followed by a 1 which may be used as a flag to detect that a character has been received. A 0 is pushed to the top of stack if there is no character in the input buffer.

b) Character Output

EMIT Removes the top of stack number and transmits it as an ASCII character to the programming terminal (Usual range 0 to 127).

If the number is greater than 127 it is transmitted as 3 ASCII characters:

ESC, Cl, C2

where Cl and C2 are the characters of low and high significance in the Hexadecimal representation of the number.

e.g. $128_{10} = 80_{16}$ is emitted as ESC 0 8

$254_{10} = FE_{16}$ is emitted as ESC E F

NUMBER Receives a number from the programming terminal and pushes it to the top of the stack. Entry of a number would normally be in response to some suitable prompt. The number must be terminated by a carriage return. Note that execution pauses while the 6433 awaits a NUMBER entry so care must be taken when using this facility.

An invalid number is not accepted, but may be edited and re-entered.

. Prints and discards the top of stack number.

." Defines the start of a print string. The string must be terminated by a " character

2.4.9 Terminal Input/Output Operators/Cont.

b) Character Output/Cont.

SPACE	Transmits a space character to the programming terminal.
CR	Transmits a carriage return character to the programming terminal.
LF	Transmits a line feed character to the programming terminal.
BS	Transmits a backspace character to the programming terminal.
NL	Transmits a new line (CR LF) character to the programming terminal.

A Print command may be executed both within a programme and immediately under Programming mode.

Note that an operator for forward space is not included in the Fixed Dictionary because no standard ASCII code exists for this function. However, it may be generated in the User Dictionary using a definition of the following form (where 9 is a commonly adopted code)

```
: FS 9 EMIT ;
```

2.4.10 Display Control

The following operators allow access to the front panel display. The ability to force the display to a selected channel (after reading the current channel if necessary) may be useful for alarm annunciation and the 8 character "Tag" display may be used for displaying relevant information.

GETBCN Returns the block and channel number of the information currently displayed on the front panel.
..... B C

SETBCN Sets the front panel display to the given block and channel number.
B C .....

TAG." Sends the ensuing text string (up to 8 ASCII characters) to the Tag display of the selected channel. The string must be terminated by a " character. If less than 8 characters are transmitted then any remaining characters are left undisturbed.

2.5 Program Control Statements

In order to start and stop program execution, skip sections of program depending upon expression values, and repeat certain statements, a number of control statements are necessary.

2.5.1 DO LOOPS

DO... LOOP

The DO and LOOP statements allow repeated execution of a block of code. DO and LOOP must always be used as a pair. The code section which they enclose can be of any length. This code is executed repeatedly; and an index value I is available.

When DO sets up the loop it takes two arguments from the stack. The top stack number is the initial index value of the loop and the second argument is the final value plus 1. If the initial value is 0 then the second argument is the number of times round the loop.

Note that loops may be nested. On execution of LOOP the stack is checked for an overflow or underflow condition.

An example is shown in section 2.7.

I

The operator I retrieves the index value of a loop and copies it onto the stack. It is useful for indexing.

LEAVE

LEAVE is used for conditional exit control from a DO LOOP. If LEAVE is executed within a loop it will set the limit to the index value, causing the loop to exit when LOOP is next executed.

DO...+LOOP

Whereas the DO...LOOP index always increments by 1, the word +LOOP allows other increments. Each time around the loop it takes the number off the stack for the increment. This increment may be computed and changed during loop execution. It may also be negative.

2.5.2 Continuous Loops

BEGIN...REPEAT

Begin starts the beginning of a continuous program loop. Only one BEGIN...REPEAT structure can be continuously executed at any time.

REPEAT terminates the BEGIN structure. On execution of REPEAT the stack is checked for an overflow or underflow condition.

A BEGIN...REPEAT structure should normally only be used within the program MAIN executed at power up.

An example is shown in section 2.7.

BEGIN...UNTIL

Whereas a BEGIN...REPEAT loop is endless, using UNTIL instead of REPEAT allows exit from the loop when a true condition is satisfied.

2.5.3 Conditional Branches

IF...ELSE...ENDIF

The IF statement destructively tests the number on the top of the stack and skips program statements up to the following ELSE or ENDIF statement.

If the top of stack number is not 0 then the statement after IF is executed. If the top of stack number is 0 then the statement after ELSE is executed instead.

ENDIF terminates an IF... structure. Every IF must have a corresponding ENDIF.

The ELSE operator is optional.

CASE...ENDCASE

The CASE structure allows branching to one of two or more possible routines depending on the value of the argument; Viz "In the Case of....do....")

The definition should always include an appropriate number of OF...ENDOF statements which define the routines corresponding to particular CASE argument values.

CASE leaves its argument at the top of the stack. This is dropped if an appropriate OF or ELSEOF statement is executed; otherwise it is dropped by ENDCASE.

2.5.3 Conditional Branches/Cont.

OF...ENDOF

The OF function defines the routine which is executed if the CASE argument is less than or equal to its own argument. If the OF condition is satisfied, the case argument is dropped and execution continues with the words following the OF. If the OF condition is not satisfied, the case argument is unaffected, and control is transferred to the word following the ENDOF. ENDOF terminates the OF definition.

It would be normal to define a series of OF statements in numerical order. Once one OF condition has been satisfied, any remaining OF statements are skipped and execution continues after the following ENDCASE.

ELSOF...ENDOF

The ELSOF statement allows for the possibility that the CASE argument may fall outside the range (i.e. greater) than the maximum OF value. In this event the ELSOF function provides a useful means of defining a default routine. It must be terminated by ENDOF in the same way as OF.

The ELSOF operator is optional.

Note that with a CASE argument of 1, the CASE... OF...ELSOF structure becomes equivalent to an IF...ELSE structure.

Examples are shown in section 2.7.

2.6 Interpreter Facilities

The 6433 interpreter incorporates a number of facilities to help the user create, test, debug and subsequently edit a program.

2.6.1 Operating Modes

The following operating modes control the execution of the interpreter.

a) Command Mode

Command mode is entered automatically when a terminal is plugged into the front panel RS232 socket and is indicated by display of ??CMD. This invites the user to enter a 2 character mnemonic in order to read and if necessary change any of the configuration parameters.

It is convenient to use a VDU to emulate the Hand Held Terminal when programming a 6433. For this purpose certain characters provide standard handheld terminal functions as follows:

Z	??CMD (Command Mode)
W	Scroll
L	Enter +
M	Enter -

Command mode may be re-entered from Programming Mode by Keying CTRL Q.

b) Program Mode

Programming Mode may be entered by keying CTRL P Access is only granted on acceptance of the valid 3 digit security code as requested by the VDU:-

```
*Please enter your security number*
```

Entering the correct code will result in the response:

```
*Hit space bar to change name*
```

If no change is required to the user name then carriage return will result in:

```
*T.C.S. Forth Version 1.(1)*
*Have a nice day*
```

Hitting the space bar will invite the user to log on with a new name up to 7 characters (including spaces) and will print a new security code which should be noted.

In Program Mode the available user memory is displayed after each entry.

2.6.1 Programming Utilities/Cont.

b) Program Mode/Cont.

If the 6433 has not yet been programmed with a valid security code then the user will be invited to log on with a suitable name the first time that Programming Mode is entered.

Continuous program execution is terminated when in Programming Mode.

Note that the VDU screen may be cleared by causing the 6433 to send a "formfeed" character using the statement

```
12 EMIT
```

c) Program Execution

A FORTH word may be executed in immediate mode by entering the word name without a colon.

Continuous Program execution is achieved by incorporating a BEGIN...REPEAT loop within a FORTH word. The reserved word MAIN would normally be used to allow automatic execution at power-up.

If a program is not being executed then the 6433 behaves just like a 6432 Signal Processor. In this mode, the input/output ranging and status may be checked out before executing a program.

When no program is being executed the message HALTED flashes in the Tag display.

Program execution may be terminated by entering an ESCAPE character or by re-entering Programming mode.

d) Escape

Entering an ESCAPE character restores the executive to normal Program Mode and clears the stack.

It is used to terminate program execution, to exit from the line editor and to recover from an error condition.

2.6.1 Operating Modes/Cont.

e) Power-Up

At power-up the user program is loaded from ROM to RAM and the executive looks for the reserved word MAIN. If this is found then this word is executed. Because program execution always uses the user dictionary in the RAM memory rather than ROM, changes can be made and validated while leaving the original program resident in ROM. Needless to say care should be taken to ensure that the updated program is stored in ROM before removing power because anything in the RAM area is over-written at power-up.

If the Word MAIN is not found then the 6433 comes up in either Command Mode or Programming Mode depending on which was last selected.

2.6.2 System Utilities

a) Terminal Configuration

WINDOW Set terminal line length

This command sets up the number of ASCII characters that may be displayed in a single line on the programming terminal. For a VDU it is normally set to 79, or 1 less than the number of characters per line to avoid automatic line feeds after the 80th position.

b) Dictionary Management

FWORDS List Word names in the fixed dictionary

This command lists the interpreter fixed dictionary words on the terminal.

UWORDS List Word names in the user dictionary

This command lists the user-created dictionary words on the terminal.

ULIST List User Program

This command provides a complete listing of the program by printing all the words in the user dictionary, in the order in which they were entered. The listing is formatted with a new line after each operator and indenting for program branches and loops etc. Entering any normal character (e.g. the space bar) may be used to interrupt the listing, and to cause it to continue subsequently.

FORGET Last Word in User Dictionary

The FORGET statement causes the last defined word to be dropped from the user dictionary. Words may only be erased on this last in first out basis since other words may be referenced by later entries.

INSERT Insert new word in user dictionary

Inserting a new word within the existing user dictionary is done by entering INSERT followed by the name of the word before which the new one is to be placed. The 6433 interpreter responds with a message that it is inserting before the desired Word, and then the new word is defined with a colon and semi-colon in the usual way. Clearly the new word may only call other words which are deeper in the dictionary.

2.6.2 System Utilities/Cont.

b) Dictionary Management/Cont.

NEW Erase RAM memory

The NEW command clears the RAM memory so as to make way for a new program. It is most useful when using an intelligent programming terminal to download a complete program, rather than simply editing an existing program when use of FORGET and INSERT is more appropriate. The program in EEPROM is undisturbed.

c) Memory

STORE Store Program from RAM to ROM

Once a program has been fully debugged and tested it should be transferred from the RAM area of the ROM/RAM emulator card to the ROM area. Note that because at power up the program in ROM is automatically transferred to the RAM, if the RAM area contains a different (i.e. edited) version then it will be overwritten.

The STORE utility takes up to about 40 seconds to execute depending on the program size. Data is verified as it is written and progress is indicated by printout of an asterisk every 100 bytes, or about one per second. If errors occur then a printout occurs of the location and both the original and incorrect data.

RECALL Recall Program from ROM to RAM

The RECALL utility transfers the content of the ROM area of the ROM/RAM emulator card into the RAM area. This is useful in reverting to a previous program after trying an edited version.

CLEAN Clean ROM Memory

This utility cleans the ROM memory area by restoring it to the unprogrammed state. It operates in a similar way to STORE by writing FFFF to every location.

d) Debugging

TRA-ON, TRA-OFF

A trace facility is available to assist in program debugging. This is enabled and disabled by the TRA-ON and TRA-OFF commands respectively. When trace is active the 6433 prints out user words as they are executed, together with the content of the stack on exit.

2.6.2 System Utilities/Cont.

d) Debugging/Cont.

Note that stack data is printed (non-destructively) in the order in which data is peeled off; i.e. Top of Stack first, at the left of the display.

2.6.3 Programming Terminal Utilities

The 6433 software provides the following utilities to allow the use of an intelligent programming terminal for program creation, editing and storage.

a) Load User Program

This command allows the user to download a program to the 6433 from a file created by an intelligent terminal. The terminal must support XON-XOFF protocol.

Load is initiated by the character STX (CTRL B). Load is terminated by the character ETX (CTRL C). This mode is protected by the security code.

Any commenting appearing within brackets is stripped off by the 6433 interpreter.

b) Save User Program

This command allows the user to dump a created program from the 6433 to a suitable storage medium.

It is similar to the command ULIST except that the user program is listed one word at a time. Save is initiated by an ENQ character (CTRL E). The 6433 transmits each Word in turn starting at the bottom of the User Dictionary. Each word is framed by an STX Start of text character (CTRL B) and an ETB End of transmission block character (CTRL W). The programming terminal must then send another ENQ character to initiate sending of the next word. Any other character terminates Save mode. The 6433 indicates the end of program by transmitting an ETX character (CTRL C) instead of ETB.

The Save utility responds to XON-XOFF protocol.

2.6.4 Program Creation

Applications programs are built up by creating a suitable structure of FORTH words within the user dictionary.

a) : NAME......... ;

A colon character defines the start of a new word in the user dictionary. A word name may consist of up to 7 ASCII characters; other characters are counted but their values are not displayed. The first word following the colon is the program name and the remaining words are the program statements. These can only consist of:-

Literals
Words from the fixed dictionary
Previously created words from the user dictionary

A semi-colon terminates the program statements begun by the colon.

Note that both the colon and semi colon must be separated from adjacent characters by a space, in the same way as any other operator.

Any words which the interpreter does not recognise will cause an error. This means that entering of a program must follow a discipline of defining the lowest level routines first and working up in program hierarchy.

Syntax errors will cause an appropriate message to be transmitted, and the offending statement will be displayed with the cursor positioned so that the error may be corrected (see section 2.6.5 on Line Editor).

A word may be examined by typing a colon followed by a space and then the word name.

The maximum length of a word is governed by the size of the Character Input buffer and the Compiler Output buffer. However, it is good programming practice to limit the word length to a single line, as set by the WINDOW function.

b) MAIN

The user word MAIN is reserved in that it defines the program statement which is automatically executed at power-up of the instrument. It normally defines the master BEGIN...REPEAT loop which schedules the program functional tasks together with any initialisation, reset etc.

2.6.4 Program Creation/Cont.

c) ERROR

The user word ERROR is reserved for execution when the executive traps on a run-time error and running of the main program is terminated.

It may be conveniently used to set outputs to a safe state or initiate an appropriate warning. The word ERROR could also be called from within the program to detect an abnormal condition such as when a time-dependent task over-runs or locks out. A common requirement would be for a "watchdog" task linked to the MAIN task scheduling loop, and updating a digital output.

2.6.5 Line Editor

The interpreter incorporates a Line Editor which allows the user to insert and delete characters in the currently displayed program statement.

The following functions are available:-

Key	Function
BACKSPACE	Move cursor back one position
TAB	Move cursor forward one position Note: This relies on tabs being set at all character positions. The tab character is defined using the SETFS command.
DELETE	Delete character at current cursor position

Characters may be inserted immediately before the current cursor position.

Note that attempting to edit the word name will create a new word, without losing the original word definition.

Editing mode may be aborted by pressing the ESCAPE key.

2.6.6 Error Reporting

Various types of errors are detected by the FORTH interpreter. These are summarised in the Programmers Reference List. In general the ESCAPE key must be pressed to recover from a programming error.

a) Syntax Errors

The interpreter checks all entries in Programming Mode for syntax errors and prints an appropriate error message. The offending statement will be displayed with the cursor positioned so that the error may be corrected. A FORTH statement will not be accepted into the user dictionary unless it is free of errors.

b) Run-time Errors

Run-time errors, such as attempting to operate on an empty stack, cause an error message to be sent to the terminal (if connected) and program execution is halted.

If the reserved word ERROR has been defined then the executive will trap on a run-time error and execute the word ERROR. This facility could be used to display an appropriate warning or set the outputs to a safe condition.

c) Halt Condition

When no program is running, which may be deliberate or after a fault condition, the message HALTED flashes on the Tag display.

2.7 Language Examples

The following figures give sample word definitions designed to illustrate the use of some of the available operators.

The first example shows a simple task scheduler which may be used in connection with the following examples. All these listings are formatted with indenting for loops etc. and each line is provided with detailed comments. Note that at least one space is required between each operator or data. This includes the colon and semi-colon characters which delimit the beginning and end respectively of the word definition. Any excess spaces are stripped off on entry. Since any text enclosed within parentheses is ignored by the 6433 interpreter the examples could be block downloaded to an instrument. For details see the 6433 Programming Terminal User Guide.

In practice an application program is likely to have a number of tasks and it is convenient to represent the configuration in diagramatic form. Appendix C shows suggested documentation for a typical application.

```
file ref PROJ003 Z011/A        memory 20 bytes plus 2 bytes per task call
5 October 1983)

: TASK1
;

: TASK2
;

: TASKn
;

: MAIN                        (Defines word MAIN, the main task scheduling loop,
                              executed at power-up)
    BEGIN                     (begin continuous loop)
      TASK1                   (Call word for Task 1)
      TASK2                   (Call word for Task 2)
                              (          :          )
      TASKn                   (Call word for Task 3)
    REPEAT                    (End of loop; loop back to BEGIN)
;                             (End of word MAIN)
```

(Note that all tasks run within MAIN must be recognised by the interpreter prior to loading MAIN; i.e. MAIN will generally be the last task in the user dictionary; hence TASK1, TASK2 etc are defined earlier)

Fig 2.1 MAIN: Simple Task Schedular

```
file ref PROJ003 Z013/A          memory  74 bytes
4 October 1983 JSH)

: CALC                 (Defines a word CALC which evaluates the
                       expression        OP1 = [ SQRT [PV1-PV2] ] / PV3
                       where Analogue inputs are on block 1
                             Analogue outputs are on block 2)
    1 1 GETAN          (Fetches the prime variable value of analogue
                       input Block 1 Channel 1, and puts it on top of the
                       stack)
    1 2 GETAN          (Fetches the PV of Block 1, Channel 2, and puts it
                       on top of the stack while the PV of Channel 1 is
                       pushed down one location)
    -                  (Subtracts the top location of the stack from the
                       second location and leaves the result PV1-PV2 on
                       top)
    SQRT               (Takes the square root of the value at the top of
                       the stack and leaves the result in its place)
    1 3 GETAN          (Fetches the PV of Block 1, Channel 3 and puts it
                       on top of the stack, pushing down the result of the
                       SQRT operation to the second location)
    /                  (Divides the value held in the second location of
                       the stack by the value on top and leaves the result
                       on top)
    2 1 SETAN          (Routes the evaluated expression on top of the
                       stack to the OP parameter of analogue output Block
                       2, Channel 1)
;                      (Defines the end of word CALC)
```

Fig 2.2   Arithmetic Computation And Signal Input/Output

Low cut off on analogue input

```
file ref PROJ003 Z007/A          memory   94  bytes
3 October 1983 JSH)

: LOWCUT                         (Implements a low cut off on an analogue
                                 input)
    1 1 GETAN                    (Gets analogue input on block 1 channel 1)
    DUP                          (Duplicates top of stack value; i.e. saves
                                 input for later)
    5 1 GETAN                    (Gets value for cut off setpoint on block 5
                                 channel 1)
    >                            (Tests whether input is greater than
                                 setpoint; leaves a 1 if true, otherwise a
                                 0)
    IF                           (True condition)
      2 1 SETAN                  (Sets output block 2 channel 1 equal to
                                 input)
    ELSE                         (False condition)
      DROP                       (Drop input value)
      0                          (Put 0 at top of stack)
      2 1 SETAN                  (Sets output block 2 channel 1 equal to 0)
    ENDIF                        (End of IF...ELSE structure)
;                                (End of word)
```

Fig 2.3 Comparison Operator and Conditional Branch:
IF...ELSE...ENDIF Structure
Low Cut Off on Analogue Input

```
file ref PROJ003 Z008/A          memory    76    bytes
3 October 1983 JSH)

: SHUNT                     (Defines a word called SHUNT to calculate the
                            value of 2 parallel resistors according to the
                            formula R1.R2/[R1+R2] R1 R2 ... R1//R2)
    2 ARG                   (Defines the top two stack entries as the 2
                            arguments R1 and R2 for the following routine.  ARG
                            also reserves an 8 entry results area at the top of
                            the stack)
    1 GETARG                (Gets thevalue of Argument 1,R1, and pushes it to
                            the top of the stack)
    2 GETARG                (Gets the valueofArgument 2,R2, and pushes it to
                            the top of the stack)
    *                       (Multiplies together the two previous values,
                            leaving the product R1.R2 at the top of the stack)
    1 GETARG                (Getsthe value of Argument 1,R1, and pushes it to
                            the top of the stack)
    2 GETARG                (Gets the value ofArgument 2,R2, and pushes it to
                            the top of the stack)
    +                       (Adds together the two previous values, leaving
                            the sum    R1+R2 at the top of the stack.
                            Note that the next value down on the stack is now
                            the product R1.R2)
    /                       (Divides the product R1.R2 by the sum R1+R2
                            leaving the
                            result)
    1 SETRES                (Defines the top of stack value as Result number
                            1)
    1 RES                   (Declares that 1 Result will be left at the top of
                            the stack) ;(Defines end of word)
```

(c.f. alternative implementation using stack manipulation operators which although more efficient is less straightforward to understand)

Fig 2.4   Local Variables: ARG...RES Structure
Computation of Value of 2 Parallel Resistors

```
file ref PROJ003 Z012/A           memory    28  bytes
4 October 1983 JSH)

: SHUNT                  (Defines a word called SHUNT to calculate the
                         value of 2 parallel resistors according to the
                         formula R1R2/[R1+R2]  R1 R2 .... R1//R2 )
   OVER                  (Brings value of R1 from second position in stack
                         and duplicates it at the top)
   OVER                  (Brings value of R2 from second position in stack
                         and duplicates it at the top)
   *                     (Multiplies top 2 stack values leaving R1.R2 at
   ROT                   (Moves value of R1 from 3rd stack position and
                         pushes onto top of stack)
   ROT                   (Moves value of R2 from 3rd stack position and
                         pushes onto top of stack)
   +                     (Adds top 2 stack values leaving R1+R2 at top)
   /                     (Divides top stack value R1+R2 by previous result
                         R1.R2)
 ;                       (End of word)

(c.f. longer but clearer implementation using ARG-RES structure)
```

Fig 2.5 Stack Manipulation Operators
Computation of Value of 2 Parallel Resistors

```
file ref PROJ003 Z009/A        memory  70 bytes
3 October 1983 JSH)

: COMBIAS                 (Defines a word to add a common bias to the first 4
                          inputs  on  block  1 and  outputs on  the  first  4
                          channels of block 2)
    5 1                   (Load  final  channel number +1 and initial channel
                          number)
    DO                    (Start DO loop)
      1                   (Load block number of An.In board)
      I                   (Get current value  I  of  index for use as channel
                          number)
      GETAN               (Get analogue input for given block/channel)
      5 1 GETAN           (Get bias value from block 5 channel 1)
      +                   (Add bias to input value)
      2                   (Load block number of An.Out board)
      I                   (Get current value I of index  for  use  as channel
                          number)
      SETAN               (Set analogue output with computed value)
    LOOP                  (End of Loop)
    ;                     (End of word)
```

FIG 2.6 DO...LOOP Structure
Adds a Common Bias to Each of 4 Inputs, Then
Retransmits on 4 Outputs

```
file ref PROJ003 Z010/A          memory    104   bytes
5 October 1983 JSH)

: SKIP                     (dummy default task)
;

: TASK1                    (dummy task 1)
;

: TASK9                    (dummy task 9)
;

: BRANCH                   (Branch routine : calls one of 9 tasks  subject
                           to the value of the input value on the stack.
                           An over-range value  skips  execution of any of the
                           tasks)
      CASE                 (Start CASE structure,  testing  value  at  top  of
                           stack)
        1
        OF                 (Test for a 1)
          TASK1            (Call task no 1)
        ENDOF
                           (       Tasks 1 to 8 as above            )
             -             (                 :       -              )
        9
        OF                 (Tests for a 9)
          TASK9            (Call task no 9)
        ENDOF
        ELSOF              (Default)
          SKIP             (Skip: default task for over-range input)
       .ENDOF
      ENDCASE              (End CASE structure)
;                          (End of word BRANCH)
```

Fig 2.7   Conditional Branch: CASE...ENDCASE Structure

2.8 Memory Requirements

The user memory of the 6433 is 4 K or 4096 bytes. This figure is reduced by several bytes because of the requirement to store system constants such as the User Name and Security Number. Available memory is listed on entering Program mode, pressing the Escape key and also after entering or editing a FORTH word.

The memory requirement in bytes of FORTH statements may be calculated using the following rules:-

a) Word Definition: Number of characters in the name string +9, rounded up to an even number of bytes

b) Literals (e.g. numerical arguments and constants): 6 bytes

c) Word Call : 2 bytes

d) Strings : Number of characters in the string +3, rounded up to an even number of bytes.

e) Operators : All 2 bytes, except for the following -

DO	2	)always used as a pair
LOOP, +LOOP	4	)
BEGIN	2	)always used as a pair
REPEAT	4	)
IF	4	)always used as a pair
ENDIF	2	)
ELSE	4	
OF	4	)always used as a pair
ENDOF	4	)

APPENDIX A. 6433 Parameter Tables

Software part No. RD 069748 issue 1, release 1

Table A.1 lists the 2 character Instrument command parameters of the 6433 Programmable Signal Processor used when accessing data via the 8260 Hand-held Terminal or the ASCII mode of the serial link protocol. The similarly accessed Channel command parameters for the 4 different types of Input/Output Blocks are given in Table A.2.

Tables A.3.1 to A.3.4 inclusive give the corresponding Parameter Numbers for the 4 types of Input/Output Blocks used with the Binary mode of the protocol.

The table below shows the modification history of the 6433 software with respect to changes in these parameter tables:-

SOFTWARE ISSUE	SOFTWARE RELEASE	MEMORY BOARD	REMARKS
1	1	MK 5	ASCII and Binary modes of the protocol supported

NOTES

The following points should be noted with regards to the tables:

TABLE A.2

(1) CN does not appear in the parameter list when accessing parameters via the serial data link. Channels are individually addressed via the link as described in the relevant sections of the Communications Manual.

(2) The 1T and 2T Channel Tag characters do not appear when the parameter list is scrolled via the (W↓) command of the 8260 Handheld Terminal. Instead, they must be accessed individually by first using the ??CMD prompt once the required Block and Channel number (CN) has been selected previously.

TABLES A.3.1 to A.3.4

* Only those parameters marked * are available with Enquiry Polling

(3) Channel Tag Characters

It should be noted that for each Channel each half of the 8 character tag is itself split into two parameters:

1T = T1 + T2

and

2T = T3 + T4

Appendix A/Cont.

(4) Instrument Parameters

II, SW, MD and those Instrument parameters relating to the Input/Output Blocks can be accessed from each of the board types.

For real blocks 1 to 4 parameter numbers 1 to 8 refer to S1 to A4 inclusive.

For pseudo blocks 5 to 8 parameter numbers 1 to 8 refer to S5 to A8 inclusive.

COMMAND MNEMONIC	COMMAND PARAMETER FUNCTION	UNITS	FORMAT	PARAMETER TYPE
II	Instrument Identity	-	5	Monitor-only
S1	Channel scan and board type	-	5	Input/Output block 1
A1	Historic alarms	-	5	
S2	Channel scan and board type	-	5	Input/Output block 2
A2	Historic alarms	-	5	
S3	Channel scan and board type	-	5	Input/Output block 3
A3	Historic alarms	-	5	
S4	Channel scan and board type	-	5	Input/Output block 4
A4	Historic alarms	-	5	
S5	Channel scan and board type	-	5	Pseudo-Input/Output block 1
A5	Historic alarms	-	5	
S6	Channel scan and board type	-	5	Pseudo-Input/Output block 2
A6	Historic alarms	-	5	
S7	Channel scan and board type	-	5	Pseudo-Input/Output block 3
A7	Historic alarms	-	5	
S8	Channel scan and board type	-	5	Pseudo-Input/Output block 4
A8	Historic alarms	-	5	
SW	Switch bank S1/S2 settings	-	5	Status words
MD	Front panel and diagnostic status indications	-	5	

TABLE A.1 List of 6433 Instrument Command Parameters and their respective mnemonics

COMMAND MNEMONIC	COMMAND PARAMETER FUNCTION	UNITS	FORMAT	PARAMETER TYPE
CN(1)	Block/Channel number	-	6	Analogue Input channels
ST	Channel status	-	5	
HR	Prime Variable high range	Eng	1	
LR	Prime Variable low range	Eng	1	
HA	High alarm (absolute)	Eng	1	
LA	Low alarm (absolute)	Eng	1	
AR	Alarm routing	-	5	
PV	Prime Variable value	Eng	1	
1T(2)	Channel Tag characters 1-4	ASCII	8	Channel Tag names
2T(2)	Channel Tag characters 5-8	ASCII	8	
CN(1)	Block/Channel number	-	6	Analogue Output channels
ST	Channel status	-	5	
HR	Prime Variable high range	Eng	1	
LR	Prime Variable low range	Eng	1	
HO	High output limit	Eng	1	
LO	Low output limit	Eng	1	
OP	Prime Variable value	Eng	1	
1T(2)	Channel Tag characters 1-4	ASCII	8	Channel Tag names
2T(2)	Channel Tag characters 5-8	ASCII	8	
CN(1)	Block/Channel number	-	6	Digital Input channels
ST	Block status	-	5	
AM	Alarm masking bits	-	5	
DS	Digital Input states	-	5	
1T(2)	Channel Tag characters 1-4	ASCII	8	Channel Tag names
2T(2)	Channel Tag characters 5-8	ASCII	8	
CN(1)	Block/Channel number	-	6	Digital Output channels
ST	Block status	-	5	
AM	Alarm masking bits	-	5	
DS	Digital Output states and enable bits	-	5	
1T(2)	Channel Tag characters 1-4	ASCII	8	Channel Tag names
2T(2)	Channel Tag characters 5-8	ASCII	8	

TABLE A.2 List of 6432 Channel Command Parameters and their respective mnemonics

		0	1	2	3	4	5	6	7	
	0	II	S1 S5	A1 A5	S2 S6	A2 A6	S3 S7	A3 A7	S4 S8	(4)
	8	A4 A8	SW	MD						(4)
CHAN 1	16	ST*	HR*	LR*	HA*	LA*	PV*	AR*		
CHAN 2	24	ST*	HR*	LR*	HA*	LA*	PV*	AR*		
CHAN 3	32	ST*	HR*	LR*	HA*	LA*	PV*	AR*		
CHAN 4	40	ST*	HR*	LR*	HA*	LA*	PV*	AR*		
CHAN 5	48	ST*	HR*	LR*	HA*	LA*	PV*	AR*		
CHAN 6	56	ST*	HR*	LR*	HA*	LA*	PV*	AR*		
CHAN 7	64	ST*	HR*	LR*	HA*	LA*	PV*	AR*		
CHAN 8	72	ST*	HR*	LR*	HA*	LA*	PV*	AR*		
CHAN 1-2	80	(3) T1	(3) T2	(3) T3	(3) T4	T1	T2	T3	T4	
CHAN 3-4	88	T1	T2	T3	T4	T1	T2	T3	T4	
CHAN 5-6	96	T1	T2	T3	T4	T1	T2	T3	T4	
CHAN 7-8	104	T1	T2	T3	T4	T1	T2	T3	T4	

TABLE A.3.1 List of 6433 Parameter Numbers, [PNO]s, and their respective mnemonics for pseudo-Analogue Input Boards

		0	1	2	3	4	5	6	7	
	0	II	S1 S5	A1 A5	S2 S6	A2 A6	S3 S7	A3 A7	S4 S8	(4)
	8	A4 A8	SW	MD						(4)
CHAN 1	16	ST*	HR*	LR*	OP*	HO*	LO*			
CHAN 2	24	ST*	HR*	LR*	OP*	HO*	LO*			
CHAN 3	32	ST*	HR*	LR*	OP*	HO*	LO*			
CHAN 4	40	ST*	HR*	LR*	OP*	HO*	LO*			
CHAN 5	48	ST*	HR*	LR*	OP*	HO*	LO*			
CHAN 6	56	ST*	HR*	LR*	OP*	HO*	LO*			
CHAN 7	64	ST*	HR*	LR*	OP*	HO*	LO*			
CHAN 8	72	ST*	HR*	LR*	OP*	HO*	LO*			
CHAN 1-2	80	(3) T1	(3) T2	(3) T3	(3) T4	T1	T2	T3	T4	
CHAN 3-4	88	T1	T2	T3	T4	T1	T2	T3	T4	
CHAN 5-6	96	T1	T2	T3	T4	T1	T2	T3	T4	
CHAN 7-8	104	T1	T2	T3	T4	T1	T2	T3	T4	

TABLE A.3.2 List of 6433 Parameter Numbers, [PNO]s, and their respective mnemonics for real and pseudo-Analogue Output Boards

		0	1	2	3	4	5	6	7	
	0	II	S1 S5	A1 A5	S2 S6	A2 A6	S3 S7	A3 A7	S4 S8	(4)
	8	A4 A8	SW	MD						(4)
CHAN 1-8	16	ST*	AM*	DS*						
	24									
	32									
	40									
	48									
	56									
	64									
	72									
CHAN 1-2	80	(3) T1	(3) T2	(3) T3	(3) T4	T1	T2	T3	T4	
CHAN 3-4	88	T1	T2	T3	T4	T1	T2	T3	T4	
CHAN 5-6	96	T1	T2	T3	T4	T1	T2	T3	T4	
CHAN 7-8	104	T1	T2	T3	T4	T1	T2	T3	T4	

TABLE A.3.3 List of 6433 Parameter Numbers, [PNO]s, and their respective mnemonics for real and pseudo-Digital Input Boards

		0	1	2	3	4	5	6	7	
	0	II	S1 S5	A1 A5	S2 S6	A2 A6	S3 S7	A3 A7	S4 S8	(4)
	8	A4 A8	SW	MD						(4)
CHAN 1-8	16	ST*	AM*	DS*						
	24									
	32									
	40									
	48									
	56									
	64									
	72									
CHAN 1-2	80	(3) T1	(3) T2	(3) T3	(3) T4	T1	T2	T3	T4	
CHAN 3-4	88	T1	T2	T3	T4	T1	T2	T3	T4	
CHAN 5-6	96	T1	T2	T3	T4	T1	T2	T3	T4	
CHAN 7-8	104	T1	T2	T3	T4	T1	T2	T3	T4	

TABLE A.3.4 List of 6433 Parameter Numbers, [PNO]s, and their respective mnemonics for real and pseudo-Digital Output Boards

CHARACTER	7-BIT BINARY CODE	HEXA-DECIMAL	DECIMAL CODE
NUL - Null	000 0000	00	0
SOH - Start of Heading	000 0001	01	1
STX - Start of Text	000 0010	02	2
ETX - End of Text	000 0011	03	3
EOT - End of transmission	000 0100	04	4
ENQ - Enquiry	000 0101	05	5
ACK - Acknowledge	000 0110	06	6
BEL - Bell	000 0111	07	7
BS - Backspace	000 1000	08	8
HT - Horizontal tabulation	000 1001	09	9
LF - Line feed	000 1010	0A	10
VT - Vertical tabulation	000 1011	0B	11
FF - Form feed	000 1100	0C	12
CR - Carriage return	000 1101	0D	13
SO - Shift Out	000 1110	0E	14
SI - Shift In	000 1111	0F	15
DLE - Data link escape	001 0000	10	16
DC1 - Device control 1	001 0001	11	17
DC2 - Device control 2	001 0010	12	18
DC3 - Device control 3	001 0011	13	19
DC4 - Device control 4 (stop)	001 0100	14	20
NAK - Negative acknowledge	001 0101	15	21
SYN - Synchronous idle	001 0110	16	22
ETB - End of Transmission Block	001 0111	17	23
CAN - Cancel	001 1000	18	24
EM - End of Medium	001 1001	19	25
SUB - Substitute	001 1010	1A	26
ESC - Escape	001 1011	1B	27
FS - File Separator	001 1100	1C	28
GS - Group Separator	001 1101	1D	29
RS - Record Separator	001 1110	1E	30
US - Unit Separator	001 1111	1F	31
DEL - Delete, Rubout	111 1111	7F	127

TABLE B.1 ASCII Control codes

CHARACTER	7-BIT BINARY CODE	HEXA-DECIMAL	DECIMAL CODE
- space	010 0000	20	32
! - exclamation mark	010 0001	21	33
" - double quotation mark	010 0010	22	34
# - hash (£ sign - ISO 7)	010 0011	23	35
$ - dollar sign (- or £ sign)	010 0100	24	36
% - percentage sign	010 0101	25	37
& - ampersand	010 0110	26	38
' - single quotation mark	010 0111	27	39
(- left-hand bracket(round)	010 1000	28	40
) - right-hand bracket(round)	010 1001	29	41
* - asterisk	010 1010	2A	42
+ - plus	010 1011	2B	43
, - comma	010 1100	2C	44
- - minus	010 1101	2D	45
. - period	010 1110	2E	46
/ - oblique	010 1111	2F	47
0 numerals	011 0000	30	48
1	011 0001	31	49
2	011 0010	32	50
3	011 0011	33	51
4	011 0100	34	52
5	011 0101	35	53
6	011 0110	36	54
7	011 0111	37	55
8	011 1000	38	56
9	011 1001	39	57
: - colon	011 1010	3A	58
; - semi-colon	011 1011	3B	59
< - less than	011 1100	3C	60
= - equals	011 1101	3D	61
> - greater than	011 1110	3E	62
? - question mark	011 1111	3F	63

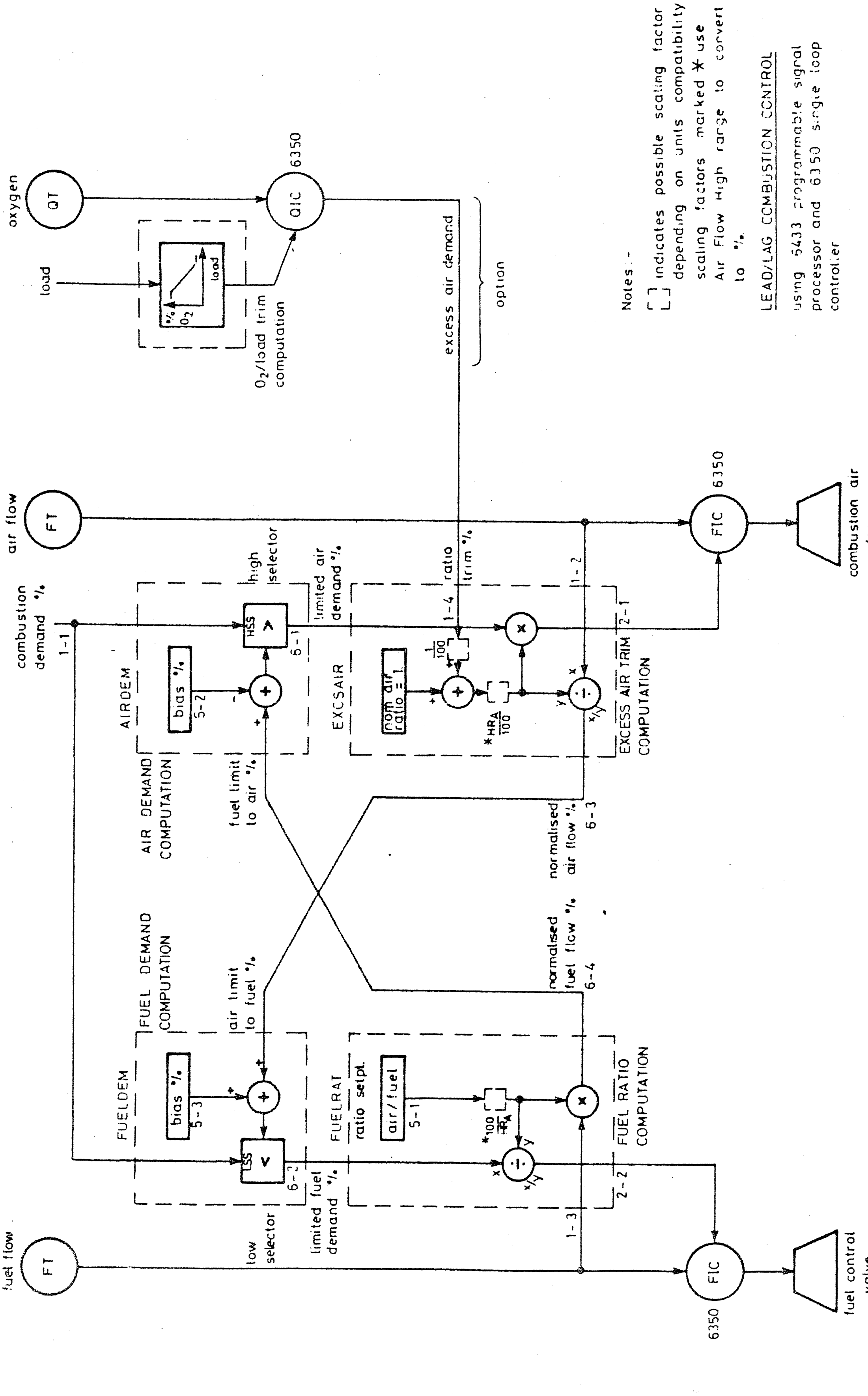
TABLE B.2 ASCII character codes

CHARACTER	7-BIT BINARY CODE	HEXA-DECIMAL	DECIMAL CODE
@ - "at" sign	100 0000	40	64
A	100 0001	41	65
B	100 0010	42	66
C	100 0011	43	67
D	100 0100	44	68
E	100 0101	45	69
F	100 0110	46	70
G	100 0111	47	71
H	100 1000	48	72
I	100 1001	49	73
J	100 1010	4A	74
K	100 1011	4B	75
L	100 1100	4C	76
M	100 1101	4D	77
N upper case letters	100 1110	4E	78
O	100 1111	4F	79
P	101 0000	50	80
Q	101 0001	51	81
R	101 0010	52	82
S	101 0011	53	83
T	101 0100	54	84
U	101 0101	55	85
V	101 0110	56	86
W	101 0111	57	87
X	101 1000	58	88
Y	101 1001	59	89
Z	101 1010	5A	90
[- LH bracket (square)	101 1011	5B	91
\ - oblique	101 1100	5C	92
] - RH bracket (square)	101 1101	5D	93
^ - up arrow (↑ common usage)	101 1110	5E	94
_ - underline(← common usage)	101 1111	5F	95

TABLE B.2 ASCII character codes

CHARACTER		7-BIT BINARY CODE	HEXA-DECIMAL	DECIMAL CODE
`	- apostrophe	110 0000	60	96
a		110 0001	61	97
b		110 0010	62	98
c		110 0011	63	99
d		110 0100	64	100
e		110 0101	65	101
f		110 0110	66	102
g		110 0111	67	103
h		110 1000	68	104
i		110 1001	69	105
j		110 1010	6A	106
k		110 1011	6B	107
l		110 1100	6C	108
m		110 1101	6D	109
n	lower case letters	110 1110	6E	110
o		110 1111	6F	111
p		111 0000	70	112
q		111 0001	71	113
r		111 0010	72	114
s		111 0011	73	115
t		111 0100	74	116
u		111 0101	75	117
v		111 0110	76	118
w		111 0111	77	119
x		111 1000	78	120
y		111 1001	79	121
z		111 1010	7A	122
{	- LH bracket (curly)	111 1011	7B	123
¦	- vertical broken line	111 1100	7C	124
}	- RH bracket (curly)	111 1101	7D	125
~	- tilde	111 1110	7E	126

TABLE B.2 ASCII character codes



Notes:-

[ ] indicates possible scaling factor depending on units compatibility; scaling factors marked * use Air Flow High range to convert to %.

LEAD/LAG COMBUSTION CONTROL

using 6433 programmable signal processor and 6350 single loop controller

Fig C.1 - Block Diagram of Lead/Lag Combustion Control

APPENDIX C Application Program, documentation example

```
COMBUSTION CONTROL WITH CROSS-LIMITING [ LEAD/LAG ]

file ref PROJ001  Z006/A     free memory 3646 bytes
13 September 1983  JSH )

: AIRDEM        (Airdem: Outputs limited air demand from the greater
                of combustion demand and fuel-limit-to-air)
      1 1 GETAN         (input combustion demand)
      6 4 GETAN         (read fuel-limit-to--air)
      5 2 GETAN         (read bias)
      -                 (subtract bias from fuel-limit-to-air)
      MAX               (select the greater)
      6 1 SETAN         (output limited air demand)
;

: FUELDEM       (Fuel demand: Outputs limited fuel demand from the
                smaller of combustion demand and air-limit-to-fuel)
      1 1 GETAN         (input combustion demand)
      6 3 GETAN         (read air-limit-to-fuel)
      5 3 GETAN         (read bias)
      +                 (add bias to air-limit-to-fuel)
      MIN               (select the smaller)
      6 2 SETAN         (output limited fuel demand)
;

: EXCSAIR       (Excess Air: Outputs air setpoint from air demand,
                as modified by excess air trim, and computes
                normalised air flow)
      1                 (read nominal air/demand ratio)
      1 4 GETAN         (input ratio trim % )
      100               (% normalising factor)
      /                 (divide by normalising factor)
      +                 (add normalised trim to nom. air/demand ratio)
      1 25 GETPAR       (get air flow High Range to normalise to %)
      100
      /                 (divide by 100)
      *                 (multiply by nett ratio)
      DUP               (save for later)
      6 1 GETAN         (read limited air demand)
      *                 (multiply by air/fuel ratio)
      2 1 SETAN         (output air flow setpoint)
      1 2 GETAN         (input measured air flow)
      SWAP              (swap top two stack entries)
      /                 (divide measured flow by nett air demand ratio)
      6 3 SETAN         (output normalised air flow)
;
```

Appendix C/Cont.

```
: FUELRAT       (Fuel ratio: outputs fuel flow setpoint and computes
                normalised fuel flow)
       5 1 GETAN        (read air/fuel ratio setpoint)
       100
       1 25 GETPAR      (get air flow High Range to normalise to %)
       /                (divide into 100)
       *                (multiply by air/fuel ratio)
       DUP              (save for later)
       6 2 GETAN        (read limited fuel demand)
       SWAP             (swop top two stack entries)
       /                (divide fuel demand by air/fuel ratio)
       2 2 SETAN        (output fuel flow setpoint)
       1 3 GETAN        (input measured fuel flow)
       *                (multiply by air/fuel ratio)
       6 4 SETAN        (output normalised fuel flow)
;

: MAIN          (Main task scheduling loop; executed at power-up)
       BEGIN            (begin loop)
         AIRDEM         (task 1: air demand computation)
         FUELDEM        (task 2: fuel demand computation)
         EXCSAIR        (task 3: excess air ratio trim)
         FUELRAT        (task 4 fuel ratio computation)
       REPEAT           (loop back to BEGIN)
;
```

APPENDIX D

First Steps in Using a 6433

1. Electrical Connections

Make connections as follows (refer to technical manual for relevant housing)
Link OV Reference to OV Power
Connect Supply - 24V DC for the 7600 BIN housing
or - AC Mains for the 7900 Powered Sleeve

2. Switch Settings

Prior to plugging in module, set up internal switches Sl and S2 (refer to 6433 facts card). In particular set up Programming Terminal Baud rate according to position of Sl no. 1:
Sl no. 1-OFF Baud Rate 300 (As for Hand held terminal)
Sl no. 1-ON Baud Rate as set on S2 no. 2, 3, 4
(Normal Value 9600 Baud with S2 no. 2,3,4 all ON)

It may be convenient to leave Sl no. 1 OFF initially to allow initial configuration via Hand held terminal.

3. Module Power Up

Plug in 6433 and switch on supply.

Observe that LEDs light on front panel. A previously unprogrammed instrument will behave functionally like a 6432.

Note that if the unit is already programmed with a word MAIN defined then program execution commences at power up. Otherwise, if no program is running, the message HALTED flashes on the Tag display.

4. 6433 Configuration

Complete a SET-UP DATA SHEET according to required database and use the hand held terminal or the programming terminal to configure the 6433 accordingly, starting with Instrument Parameters.

> Note: Display of a Hardware error may indicate incorrect setting of Board type parameters Sl to S8 Display of a Sumcheck Error may indicate incompatible High and Low ranges or Setpoint/Output limits. Reset Channel Parameters ST and Instrument Status word MD by entering 0000 if necessary.

APPENDIX D/Cont.

5. Programming Mode

Connect Programming Terminal to front panel socket ensuring that Baud Rate on the instrument is set correctly as paragraph 2.

Note that the Programming Terminal must be set up for the correct data format etc. Also the line length card and tab character (to be used as a forward space in editing) must be set correctly in the 6433. Usually the default values will be adequate.

Check emulation of the hand held terminal by entering Z for Command mode as indicated by ??CMD.

All entries are terminated by RETURN.

Enter CTRL P to select Programming Mode. Note that command mode may be re-entered using CTRL Q. Sign on as requested, entering either a user name (7 characters including spaces) with a previously unprogrammed instrument or the 3 digit security code if a user name is already programmed. Note that instruments supplied pre-programmed by TCS have the user name TCS (preceded by a space to make up to 7 characters) for which the security code is 404. Hit the Space Bar to change name if required; otherwise press RETURN. The 6433 will then print available user memory.

Check Fixed Dictionary words by entering FWORDS
Check User Dictionary words by entering UWORDS

6. Executing Forth Statements in Immediate Mode

Try the features of the FORTH interpreter by entering simple arithmetic statements followed by a dot so as to print the result. Note that all operators must be separated by a space.

E.g. 50 2 / 40 * 1 - . RETURN

(Prints the result of the calculation [(50/2) x 40] -1 i.e. 999)

3 ." COME IN No " . RETURN

(Prints COME IN No 3)

Remember that FORTH statements are entered onto the stack in Reverse Polish Format. All entries must be separated by a space.

APPENDIX D/Cont.

7. Editing a FORTH Word

A word may be listed by entering a colon followed by a space and then the word name.

Editing may then proceed in an obvious way using BACKSPACE and DELETE keys. Usually the TAB key is used for forward space.

8. Entering a Simple Program

Try defining a user word by entering a colon followed by a space and then the word name and then a set of operations, always terminated by a space and a semi-colon. The preceding numerical example may be coded as a FORTH word TEST1 :

```
: TEST1 50 2 * 40 * 1 - . ;
```

The last word entered in the User Dictionary may be erased by typing FORGET.

9. Listing a Program

The complete program may be listed using the ULIST command, using the space bar to page through the listing.

10. Running a Program

To execute TEST1, simply type:

```
TEST1    RETURN .
```

To execute TEST1 continuously in immediate mode try a continuous loop

```
BEGIN TEST1  NL REPEAT
```

Execution may be terminated by pressing ESCAPE .

Now add a second word TEST2 :

```
: TEST2 3 ." COME IN No" . NL ;
```

Try a single execution, then create a new word RUN to run both TEST1 and TEST2 consecutively.

```
: RUN BEGIN TEST1 NL TEST2 REPEAT
```

APPENDIX D/Cont.

11. Fixing a Program

The program may be fixed in ROM using the STORE command. Try this but first check the User Dictionary by entering UWORDS.

Now enter STORE. The program will now be retained even when the instrument is unpowered.

Now erase the User Dictionary by entering NEW or by using the FORGET command as many times as necessary.

UWORDS should now show an empty dictionary.

If the instrument supply is now removed and re-connected the original program will be reloaded from ROM to RAM. This may be checked using UWORDS.

12. Execution and Power Up

Automatic execution at power up may be achieved by incorporating the program within the word MAIN. If a MAIN word has been defined then the instrument will come up in command mode, allowing access via the hand held terminal. Otherwise it will come up in either command or programming mode according to the previously selected mode.

APPENDIX E

6433 PROGRAMMER REFERENCE LIST

Stack inputs and outputs are shown with top of stack on right.

Operand key: n,nl represents a 32 bit signal floating point number.

Specific types are distinguished as follows:

Bn	Input/Output Block Number
Cn	Input/Output Channel Numbers
Pn	Parameter Number
Vn	Variable Number
An	Argument Number
Rn	Result Number
Tn	Timer Number
f	Boolean Flag
c	ASCII Characters

ARITHMETIC

+
-
*
/
ABS
MAX
MIN
MINUS
SQR
SQRT
SIN
COS
ATAN
ATAN2
E
EXP
LN
PI
INT

LOGICAL

AND
OR
XOR
NOT

COMPARISON

>
<
=
0>
0<

APPENDIX E/Cont.

INPUT/OUTPUT

GETAN
SETAN
GETDIG
SETDIG
GETPAR
SETPAR

GLOBAL VARIABLE

GETVAR
SETVAR
DIFVAR
SUMVAR

LOCAL VARIABLE

GETARG
SETARG
GETRES
SETRES

TIMER

GETTIM
SETTIM
+TIM

STACK MANIPULATION

DROP
DUP
OVER
ROT
SWAP
ARG
RES

TERMINAL INPUT/OUTPUT

KEY
NUMBER
EMIT
."
.
SPACE
CR
LF
BS
NL

APPENDIX E/Cont.

DISPLAY CONTROL

GETBCN
SETBCN
TAG."

CONTROL STRUCTURES

DO...LOOP
I
LEAVE
DO...+LOOP
BEGIN...REPEAT
BEGIN...UNTIL
IF...ENDIF
ELSE
CASE...ENDCASE
OF...ENDOF
ELSOF...ENDOF

SYSTEM UTILITIES

WINDOW
FWORDS
UWORDS
ULIST
FORGET
INSERT
NEW
STORE
RECALL
CLEAN
TRA-ON
TRA-OFF

PROGRAM CREATION

:
;
MAIN
ERROR

LINE EDITOR

BACKSPACE
TAB
DELETE

APPENDIX E/Cont.

OPERATING MODES

Command Mode	:	DC1 (CTRL Q)
??CMD	:	Z
Scroll	:	W
Enter +	:	L
Enter -	:	M
Programming Mode	:	DLE (CTRL P)
Program Execution	:	Word MAIN is executed at power up. Word ERROR is executed on trap to Run-time error
Termination	:	ESCAPE terminates program execution or Edit mode and resets error condition.

PROGRAMMING TERMINAL UTILITIES

CHARACTER	UTILITY	
DC1 (CTRL Q) DLE (CTRL P) ESCAPE	Command Mode Programming Mode Terminate execution	Operating Modes
STX (CTRL B) ETX (CTRL C)	Initiate Terminate	Load
ENQ (CTRL E) STX (CTRL B) ETB (CTRL W) ETX (CTRL C)	Initiate Start of Text ⎫ from End of Block ⎬ 6433 End of Text ⎭	Save

ERROR MESSAGES

Matching pairs

Compiler output buffer overflow

Terminal input buffer overflow

Data stack overflow

Data stack underflow

Attempting to edit a FIXED word

I/O board type

LOCAL VARIABLE argument out of range

TIMER argument out of range

ARG/RES argument out of range

I/O Bn out of range

I/O Cn out or range

Undefined or forward referenced word

I/O Pn argument out of range

User dictionary empty

Memory corruption

VARIABLE argument out of range

I/O Write protected

I/O Board hardware

User memory not available

Missing ARG statement

Illegal word use

Word sumcheck

BOARD CODE 00	BOARD TYPE AN. IN											
BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OR HA/HQ	LOW ALM/OR LA/LQ	ALM ROUTING AR/- ALM MASKING AM/AM	PRIME VAR PV/OR DIG STATUS DS/DS	CHANNEL TAG CHAR 1T	2T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
11	2000	99.99	00.00	99.99	00.00	0000		COMB	DEM	°/o		COMBUSTION DEMAND
12	0001	9999	0000	9999	0000	0000		AIR	FLOW	M³/H		AIR FLOW
13	0001	2000	0000	2000	0020	0041		FUEL	FLOW	M³/H		FUEL FLOW
14	2000	10.00	10-00	10.00	10-00	0000		RAT	TRIM	°/o		RATIO TRIM
15												
16												
17												
18												

BOARD CODE 08	BOARD TYPE AN. OUT											
BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/ ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1T	2T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
21	0000	9999	0000	9999	0000			AIR	DEM	M³/H		AIR DEMAND
22	0000	2000	0000	2000	0000			FUEL	DEM	M³/H		FUEL DEMAND
23												
24												
25												
26												
27												
28												

BOARD CODE 10	BOARD TYPE DIG. IN											
BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/ ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1T	2T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
31	0000					00C0		AIR	LOW			AIR PRESSURE LOW
32								FUEL	LOW			FUEL PRESSURE LOW
33												
34												
35												
36												
37												
38												

BOARD CODE 18	BOARD TYPE DIG. OUT											
BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/ ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1T	2T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
41	0000					0080		FUEL	ALM			FUEL LOW ALARM
42												
43												
44												
45												
46												
47												
48												

□ = READ ONLY

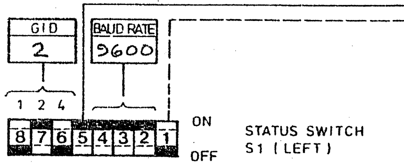
TURNBULL CONTROL SYSTEMS LTD

6433 SIGNAL PROCESSOR CHANNEL PARAMETERS

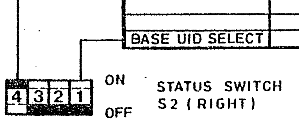
SERIAL COMMS

* DELETE AS APPLICABLE

GID	BAUD RATE
2	9600



STATUS SWITCH
S1 (LEFT)



STATUS SWITCH
S2 (RIGHT)

MODE SELECTION

FUNCTION	OFF *	ON *
PROTOCOL MODE	~~ASCII~~	BINARY
TERMINAL BAUD RATE	300	~~SW.SEL.~~
TAG DISPLAY	~~DISABLE~~	ENABLE
BASE UID SELECT	0	~~8~~

BOARD FUNCTION : REAL INPUT / OUTPUT CHANNELS

SIGNAL TYPES								
BOARD CODE/TYPE	00 ANALOGUE IN		08 ANALOGUE OUT		10 DIGITAL IN		18 DIGITAL OUT	
CHANNEL	BLOCK 1	CHAN ACTIVE	BLOCK 2	CHAN ACTIVE	BLOCK 3	CHAN ACTIVE	BLOCK 4	CHAN ACTIVE
1	0-10V		0-10V		0/15V		0/15V	
2	1-5V		0-10V		0/15V			
3	1-5V							
4	±5V(WRT+5V)							
5								
6								
7								
8								

INSTRUMENT PARAMETERS

PARAMETER TYPE	PARAMETER FUNCTION	MNEMONIC	VALUE (4 CHARS)	UNITS
BLOCK 1	CHANNEL SCAN + BOARD TYPE	S 1	0400	
	HISTORIC ALARMS	A 1		
BLOCK 2	CHANNEL SCAN + BOARD TYPE	S 2	0208	
	HISTORIC ALARMS	A 2		
BLOCK 3	CHANNEL SCAN + BOARD TYPE	S 3	0210	
	HISTORIC ALARMS	A 3		
BLOCK 4	CHANNEL SCAN + BOARD TYPE	S 4	0118	
	HISTORIC ALARMS	A 4		

CONTINUED ON SHEET 2

□ = READ ONLY OR READ / WRITE

ISS.	DATE	COMPILED	INSTRUMENT IDENTITY II	FUNCTION
A	23·1·84	JSH	6433/1	LEAD/LAG COMBUSTION CONTROL
		CHECKED	SERIAL NO. INT 082/1/1/8/83	

TURNBULL CONTROL SYSTEMS LTD

6433 PROGRAMMABLE SIGNAL PROCESSOR SET UP DATA SHT

SHT. 1 OF 2.

BOARD CODE 08 | BOARD TYPE AN. OUT OPERATOR SET

BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/- ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1T	CHANNEL TAG CHAR 2T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
51	2000	99.99	00.00	10.00	00.00		05.00	A-F	BIAS	%		AIR LIM TO FUEL BIAS
52	2000	99.99	00.00	10.00	00.00		05.00	F-A	BIAS	%		FUEL LIM TO AIR BIAS
53												
54												
55												
56												
57												
58												

BOARD CODE 00 | BOARD TYPE AN. IN DERIVED VALUES

BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/- ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1T	CHANNEL TAG CHAR 2T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
61	2000	99.99	00.00	99.99	00.00	0000		AIR	DEM%	%		LIMITED AIR DEMAND
62	2000	99.99	00.00	99.99	00.00	0000		FUEL	DEM%	%		LIMITED FUEL DEMAND
63	2000	99.99	00.00	99.99	00.00	0000		NORM	AIR	%		NORMALISED AIR FLOW
64	2000	99.99	00.00	99.99	00.00	0000		NORM	FUEL	%		NORMALISED FUEL FLOW
65												
66												
67												
68												

BOARD CODE | BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/- ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1T	CHANNEL TAG CHAR 2T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
71												
72												
73												
74												
75												
76												
77												
78												

BOARD CODE | BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/- ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1T	CHANNEL TAG CHAR 2T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
81												
82												
83												
84												
85												
86												
87												
88												

□ = READ ONLY

TURNBULL CONTROL SYSTEMS LTD | 6433 SIGNAL PROCESSOR CHANNEL PARAMETERS

PROGRAMMING TERMINAL SET-UP

FUNCTION	STATE
SCROLL STYLE	JUMP
AUTO REPEAT	ON
AUTO WRAPAROUND	ON
NEW LINE	OFF
INTERLACE	OFF
ANSI	YES
AUTO X ON-X OFF	OFF
PARITY (ENABLED)	EVEN
BITS / CHAR	7
STOP / BITS	1

FUNCTION	VALUE	6433 WORD	VALUE
LINE LENGTH	80	WINDOW	79
BAUD RATE	9600		

DEC VT100 SETUP :

1				2				3				4			
0	1					1	0		1	0	0	1	1	0	1

BOARD FUNCTION : INTERNAL 'PSEUDO' CHANNELS

BOARD CODE/TYPE								
CHANNEL	BLOCK 5	CHAN ACTIVE	BLOCK 6	CHAN ACTIVE	BLOCK 7	CHAN ACTIVE	BLOCK 8	CHAN ACTIVE
1								
2								
3								
4								
5								
6								
7								
8								

INSTRUMENT PARAMETERS

PARAMETER TYPE	PARAMETER FUNCTION	MNEMONIC	VALUE (4 CHARS)	UNITS
BLOCK 5	CHANNEL SCAN BOARD TYPE	S 5	0208	▨
	HISTORIC ALARMS	A 5	▨	▨
BLOCK 6	CHANNEL SCAN BOARD TYPE	S 6	0400	▨
	HISTORIC ALARMS	A 6	▨	▨
BLOCK 7	CHANNEL SCAN BOARD TYPE	S 7	0000	▨
	HISTORIC ALARMS	A 7	▨	▨
BLOCK 8	CHANNEL SCAN BOARD TYPE	S 8	0000	▨
	HISTORIC ALARMS	A 8	▨	▨
MONITOR ONLY	SWITCH BANK S1 & S2 SETTINGS	S W	▨	▨
	6433 OPERATING STATUS	M D	▨	▨

▨ = READ ONLY OR READ / WRITE

ISS.	DATE	COMPILED	PROGRAM FILE REF.	FUNCTION
A	23.1.84	JSH	2006	LEAD/LAG COMBUSTION CONTROL
		CHECKED		

TURNBULL CONTROL SYSTEMS LTD

6433 PROGRAMMABLE SIGNAL PROCESSOR SET UP DATA SHT

SHT. 2 OF 2.

BOARD CODE	BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM / OP LA/LO	ALM ROUTING AR/ -	PRIME VAR PV/OP	CHANNEL TAG CHAR		UNITS	AN IN/OUT	FUNCTION
	BLOCK STATUS ST	NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT				ALM MASKING AM/AM	DIG STATUS DS/DS	1 T	2 T	N/A	DIG IN/OUT	
51												
52												
53												
54												
55												
56												
57												
58												

BOARD CODE	BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/ -	PRIME VAR PV/OP	CHANNEL TAG CHAR		UNITS	AN IN/OUT	FUNCTION
	BLOCK STATUS ST	NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT				ALM MASKING AM/AM	DIG STATUS DS/DS	1 T	2 T	N/A	DIG IN/OUT	
61												
62												
63												
64												
65												
66												
67												
68												

BOARD CODE	BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/ -	PRIME VAR PV/OP	CHANNEL TAG CHAR		UNITS	AN IN/OUT	FUNCTION
	BLOCK STATUS ST	NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT				ALM MASKING AM/AM	DIG STATUS DS/DS	1 T	2 T	N/A	DIG IN/OUT	
71												
72												
73												
74												
75												
76												
77												
78												

BOARD CODE	BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/OP LA/LO	ALM ROUTING AR/	PRIME VAR PV/OP	CHANNEL TAG CHAR		UNITS	AN IN/OUT	FUNCTION
	BLOCK STATUS ST	NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT				ALM MASKING AM/AM	DIG STATUS DS/DS	1 T	2 T	N/A	DIG IN/OUT	
81												
82												
83												
84												
85												
86												
87												
88												

□ = READ ONLY

TURNBULL CONTROL SYSTEMS LTD

6433 SIGNAL PROCESSOR CHANNEL PARAMETERS

PROGRAMMING TERMINAL SET-UP

FUNCTION	STATE
SCROLL STYLE	JUMP
AUTO REPEAT	ON
AUTO WRAPAROUND	ON
NEW LINE	OFF
INTERLACE	OFF
ANSI	YES
AUTO X ON - X OFF	OFF
PARITY (ENABLED)	EVEN
BITS / CHAR	7
STOP / BITS	1

FUNCTION	VALUE	6433 WORD	VALUE
LINE LENGTH		WINDOW	
BAUD RATE			

DEC VT100 SETUP :

1				2				3				4			
0	1					1	0		1	0	0	1	1	0	1

BOARD FUNCTION : INTERNAL 'PSEUDO' CHANNELS

BOARD CODE/TYPE								
CHANNEL	BLOCK 5	CHAN ACTIVE	BLOCK 6	CHAN ACTIVE	BLOCK 7	CHAN ACTIVE	BLOCK 8	CHAN ACTIVE
1								
2								
3								
4								
5								
6								
7								
8								

INSTRUMENT PARAMETERS

PARAMETER TYPE	PARAMETER FUNCTION	MNEMONIC	VALUE (4 CHARS)	UNITS
				▨
BLOCK 5	CHANNEL SCAN BOARD TYPE	S 5	0_ _ _	▨
	HISTORIC ALARMS	A 5	▨	▨
BLOCK 6	CHANNEL SCAN BOARD TYPE	S 6	0_ _ _	▨
	HISTORIC ALARMS	A 6	▨	▨
BLOCK 7	CHANNEL SCAN BOARD TYPE	S 7	0_ _ _	▨
	HISTORIC ALARMS	A 7	▨	▨
BLOCK 8	CHANNEL SCAN BOARD TYPE	S 8	0_ _ _	▨
	HISTORIC ALARMS	A 8	▨	▨
MONITOR ONLY	SWITCH BANK S1 & S2 SETTINGS	SW	▨	▨
	6433 OPERATING STATUS	MD	▨	▨

▨ = READ ONLY OR READ / WRITE.

ISS.	DATE	COMPILED	PROGRAM FILE REF.	FUNCTION
		CHECKED		

TURNBULL CONTROL SYSTEMS LTD

6433 PROGRAMMABLE SIGNAL PROCESSOR SET UP DATA SHT

BOARD CODE	BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/ OP LA/LO	ALM ROUTING AR/ - ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1 T	2 T	UNITS N/A	AN IN/OUT DIG IN/ OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
11												
12												
13												
14												
15												
16												
17												
18												

BOARD CODE	BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/ OP LA/LO	ALM ROUTING AR/ - ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1 T	2 T	UNITS N/A	AN IN/OUT DIG IN/ OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
21												
22												
23												
24												
25												
26												
27												
28												

BOARD CODE	BOARD TYPE

BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/ OP LA/LO	ALM ROUTING AR/ - ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1 T	2 T	UNITS N/A	AN IN/OUT DIG IN/ OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
31												
32												
33												
34												
35												
36												
37												
38												

BOARD CODE	BOARD TYPE

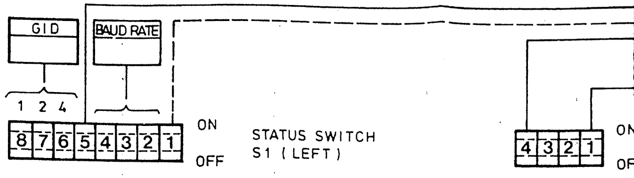
BLOCK / CHN CN	CHAN STATUS ST BLOCK STATUS ST	HIGH RANGE HR	LOW RANGE LR	HIGH ALM/OP HA/HO	LOW ALM/ OP LA/LO	ALM ROUTING AR/ ALM MASKING AM/AM	PRIME VAR PV/OP DIG STATUS DS/DS	CHANNEL TAG CHAR 1 T	2 T	UNITS N/A	AN IN/OUT DIG IN/OUT	FUNCTION
		NOT APPLICABLE FOR DIGITAL INPUT / OUTPUT										
41												
42												
43												
44												
45												
46												
47												
48												

□ = READ ONLY

TCS TURNBULL CONTROL SYSTEMS LTD | EI | 6433 SIGNAL PROCESSOR CHANNEL PARAMETERS

SERIAL COMMS

* DELETE AS APPLICABLE



MODE SELECTION

FUNCTION	OFF *	ON *
PROTOCOL MODE	ASCII	BINARY
TERMINAL BAUD RATE	300	SW.SEL.
TAG DISPLAY	DISABLE	ENABLE
BASE UID SELECT	0	8

BOARD FUNCTION : REAL INPUT / OUTPUT CHANNELS

BOARD CODE/TYPE								
CHANNEL	BLOCK 1	CHAN ACTIVE	BLOCK 2	CHAN ACTIVE	BLOCK 3	CHAN ACTIVE	BLOCK 4	CHAN ACTIVE
1								
2								
3								
4								
5								
6								
7								
8								

INSTRUMENT PARAMETERS

PARAMETER TYPE	PARAMETER FUNCTION	MNEMONIC	VALUE (4 CHARS)	UNITS
				▨
BLOCK 1	CHANNEL SCAN + BOARD TYPE	S 1	0___	▨
	HISTORIC ALARMS	A 1	▨	▨
BLOCK 2	CHANNEL SCAN + BOARD TYPE	S 2	0___	▨
	HISTORIC ALARMS	A 2	▨	▨
BLOCK 3	CHANNEL SCAN + BOARD TYPE	S 3	0___	▨
	HISTORIC ALARMS	A 3	▨	▨
BLOCK 4	CHANNEL SCAN + BOARD TYPE	S 4	0___	▨
	HISTORIC ALARMS	A 4	▨	▨
	CONTINUED ON SHEET 2			

▨ = READ ONLY OR READ / WRITE

ISS.	DATE	COMPILED	INSTRUMENT IDENTITY II 6433/1	FUNCTION
		CHECKED	SERIAL NO.	

TCS TURNBULL CONTROL SYSTEMS LTD | EI | 6433 PROGRAMMABLE SIGNAL PROCESSOR SET UP DATA SHT