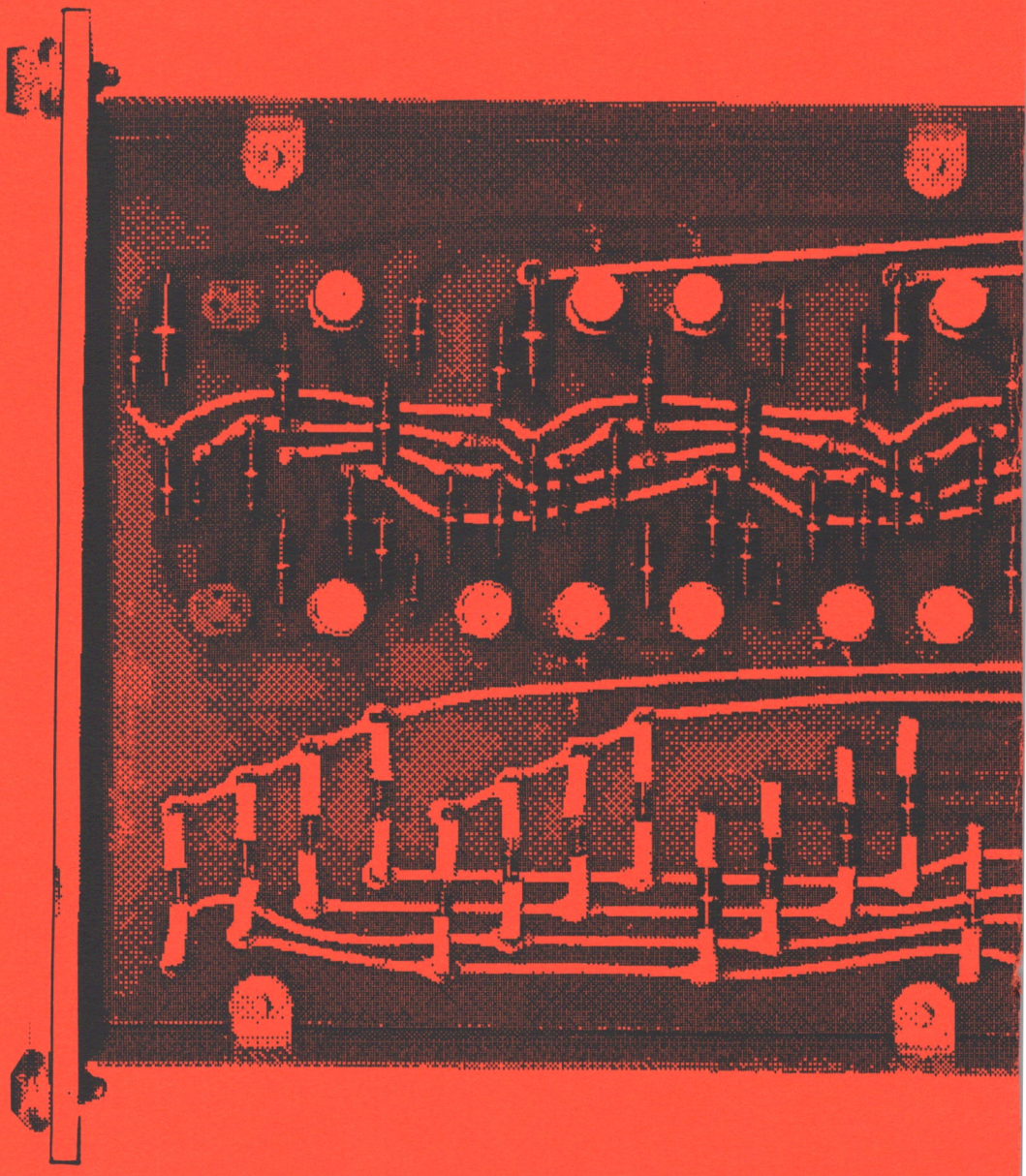
ACMS' Bailey 756 multiprocessor

by John Deane

AUSTRALIAN COMPUTER MUSEUM SOCIETY



Cover: ***composite of Bailey cabinets***

Inside: ***a Bailey slide-in logic module***

BAILEY METER 756

(& 721)

Acknowledgements

My thanks to Eric Werme, son of the 756's designer John Werme, for excellent background help and photos of his father. Many thanks also to Dick Browngardt and Ron Brayan of Elsag Bailey Australia for details of Bailey systems development and information about Bailey in Australia.

My further thanks to Elsag Bailey for permission to use their technical illustrations on pages 11, 17, 18, 36, 37; to Ron Brayan for permission to use his photos on pages 7, 8, 19 and the back cover; and to ETSA for permission to use their photos on pages 4 and 6.

Copyright © 1999 John Deane
jdeane@tip.csiro.au
Published by

The Australian Computer Museum Society Inc.
PO Box 103, Killara NSW 2071

The computer industry is now old enough to have a history. If we do not preserve that history it will disappear forever.

Control, Alt, Preserve

ISBN 0 9585678 1 6

CONTENTS

Introduction

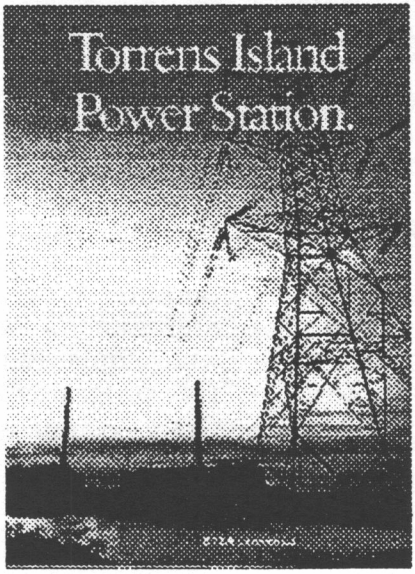
The ACMS has a Bailey 756 computer - 8 tons of transistor based, 1962 vintage hardware housed in 16 nuclear attack proof racks.

What it ...

IS	*ISN'T*
A digital computer.	A process control system.
A parallel processing system.	An interactive system.
A mixed analog & digital system.	Binary.
A (mostly) special purpose system.	Simple.
A drum memory computer.	Fast.

Well ... three racks of it is a Bailey 721 which *is* a process control system - and all analog too!

It came with enough documentation to recreate one from raw materials *and* a cubic metre of spares. What it *didn't* come with was its main peripheral:



Development

John V. Werme designed the 756 and was lead engineer for its development.

... I worked for [him] when I came into Cleveland for training. A wonderful smart and very objective person.

- Dick Browngardt

My father was very fond of that machine... Dad claimed the 756 was so easy to program that a 12 year old could do it, then proceeded to teach me how.

- Eric Werme

John Werme had experience with vacuum-tube electronics at Philco and Honeywell Industrial Control before he joined Metrotype Corp. (of Michigan City, Indiana) in 1956. The next year Bailey Meter expanded into digital control by purchasing Metrotype and John formed a group to exploit new technology and design a range of *transistorised* control modules. Between 1960 and 1974 they developed these into digital information systems, sequence controllers, analog control systems and a range of minicomputers. The systems included:

720 series of analog electronic control systems,
750 series data loggers:
755 which included a Packard Bell computer,
756 used a computing subsystem made up of Bailey modules,
760, a subset of the 750, for "hardwired" control,
820 which used integrated circuits and replaced the 720s,
855 with integrated circuits and a core memory, it replaced the 756.

John was granted patent #3,266,023 for fundamental developments in parallel systems. He followed this with a new range of plant

instrumentation and progressed to the top of Bailey's engineering divisions. In 1975 John Werme left Bailey.

Bailey in Australia

South Australia's Torrens Island Power Station bought the first 721 System in Australia. It was built in the USA but quite a number of subsequent 721 (and its replacement, the 820) controllers were configured and built by Bailey Australia.

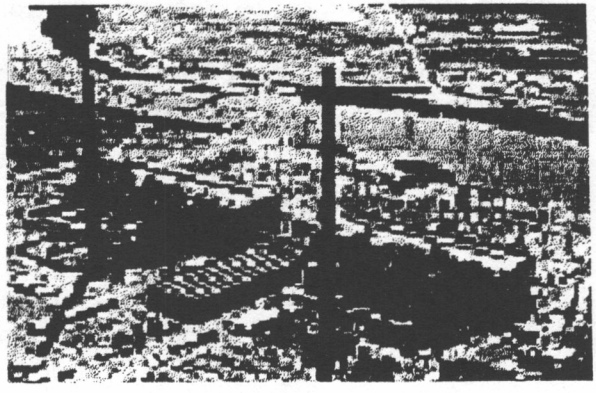
One 750 data logger was designed and built in Australia for shipboard use without a magnetic drum. The system did scanning and alarm monitoring using banks of thumbwheel switches as its memory.

The Torrens Island 756 system was also imported from the USA, but a second one was built in Australia for a power station in Victoria.

Torrens Island

About 1962 the Bailey Meter Co. (of Wickliffe Ohio) completed their 756 computer as part of their power station monitoring product line.

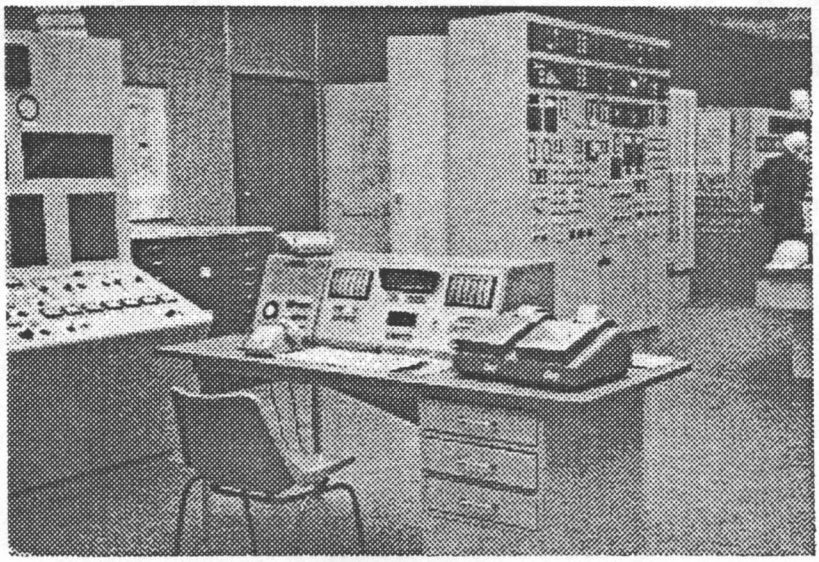
Around the same time the Electricity Trust of South Australia (ETSA) was building a power station at Torrens Island north of Adelaide. They chose Bailey and turned their 756 system on in 1965. The whole station went live in 1967.



Torrents Island Power Station - the buildings on the right housed the "A" Power Station and the 756

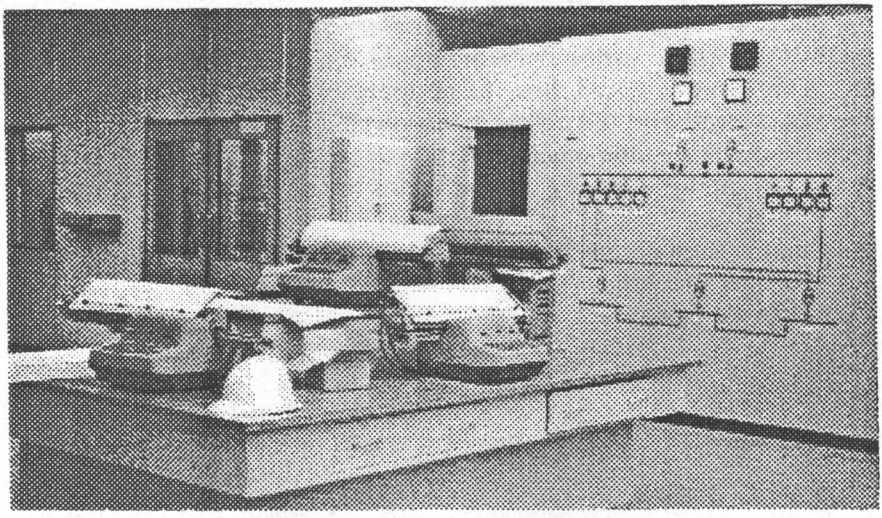
The job of the 756 was to monitor plant performance and help the station conserve fuel oil (which was the main cost) - in addition to logging everything. There were a number of on line performance programs used to calculate boiler efficiency and do some heat balances that could not otherwise be measured. ETSA employed a *Deviation Concept Monitor* to help the operator improve the overall efficiency of the plant. Based on the design parameters of the plant, key targets were established which varied over the operating load of the boiler, turbine, etc. The 756 calculated these ideal operating points for the 721 analog control system (and the operator) to try and achieve. The actual measured target value was compared to the ideal and any deviation shown on the little analog indicators on the operator station including

- main steam temperature,
- turbine vacuum,
- throttle pressure
- reheat steam temperature,
- sootblower steam flow,
- etc (11 in all)



Control desk with Clary printers

This provided a challenge to the operator's ability to offset a negative efficiency variation with a positive one from another control point. Every fifteen minutes the logger would print out the dollar effect of the overall summary of deviations: in RED if the performance was costing more money due to operations.



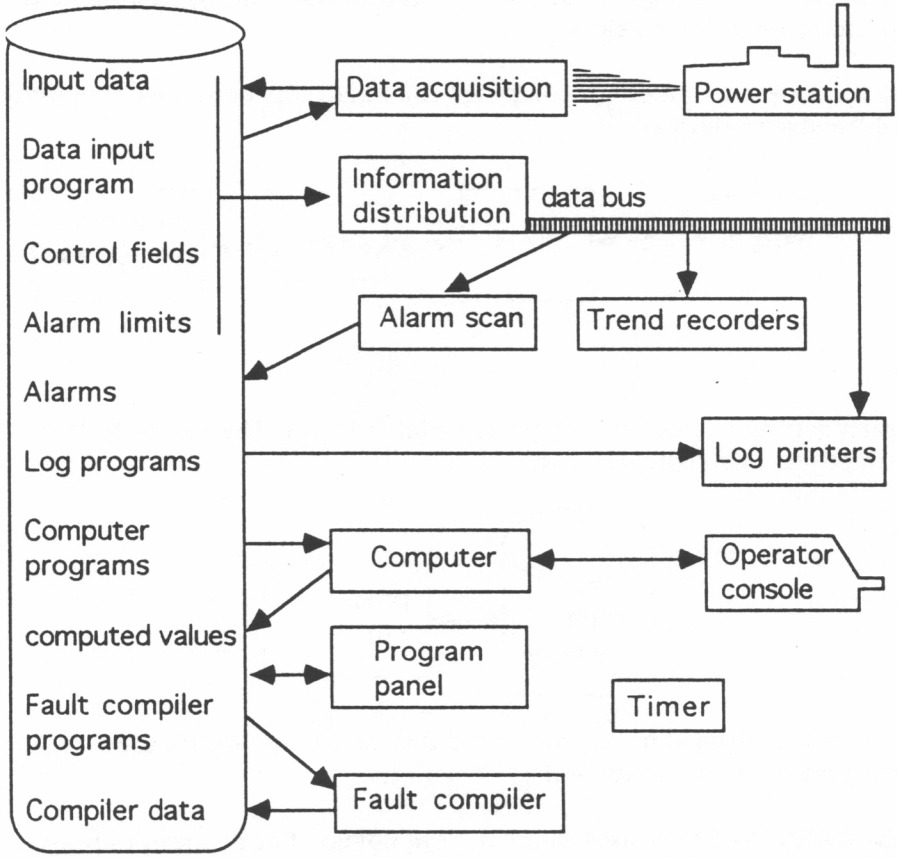
The logging printers

The 756 did all this for some 30 years before it was turned off ... and Ken Kirkby (of *Real Time Hardware & Software*) noticed. He mentioned this to ACMS' John Geremin who got in touch with Alex Ashley (of ETSA's inheritor *Optima Energy*), Matthew Connell (of the Powerhouse Museum), the History Trust of South Australia and Ron Brayan (of *Elsag Bailey* in Sydney). The final outcome was that ACMS became the proud owners of the Bailey equipment. John came to an arrangement with *Verdon Transport* and the cabinets came up to Elsag Bailey's basement (with John nursing the fragile drum), then to ACMS storage at Homebush in early 1999.

The Hardware:

Memory

A common memory binds the parallel sub-systems together and makes them behave as one. The memory *is* a rotating magnetic coated cylindrical drum; a Bryant model 7514 rotating at 1500 rpm. It has predefined functional areas which seem like a directory but they are hard-wired into the sub-systems.



Memory **Parallel systems in the 756**

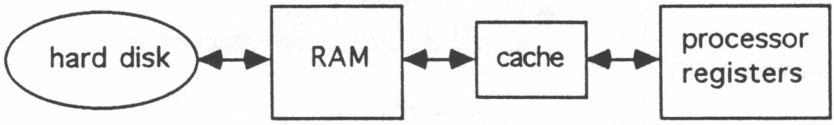
These blocks show some of the sub-systems which operate in parallel.

A number of these are programmable: not only the computer itself, but the data acquisition system *and* the fault compiler *and* the log printers *and* the alarm scan - to varying degrees.

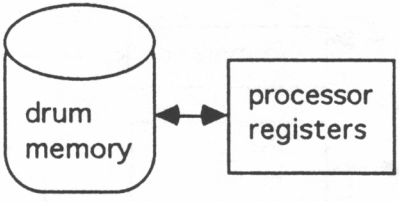
Other systems such as the information distribution system, timer and program panel have hard-wired operating logic.

These semi-independent parallel units are matched by a very parallel memory - it could do simultaneous data transfers using its 280 fixed position read/write heads.

If you look into a modern computer you might expect to see a hierarchy of "memory" devices



each getting smaller and faster from left to right. The Bailey system simplifies this right down to



where the drum is quite like a hard disk and there are multiple processors! Now for some BAILEY definitions.

⌦ WARNING Bailey uses familiar terminology, but it doesn't always mean the same thing.

WORD This was what the systems worked with and the fundamental

storage on the drum. A WORD was a signed 6 digit number held in 25 bits: one bit for sign then 6 groups of 4 bits, each representing one digit .

It's a ***decimal*** machine, ***not binary.***

eg 756 was stored as +000756 or in bits

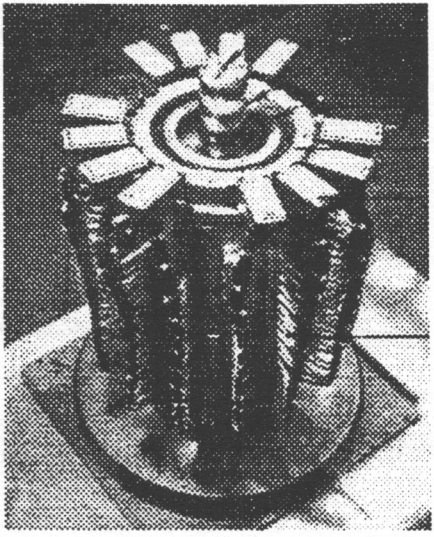
```
  0 0000 0000 0000 0111 0101 0110
S=+ S5=0 S4=0 S3=0 S2=7 S1=5 S0=6    digit identifiers
 24.............................0    bits
```

⌦ SECTOR A SECTOR is the smallest addressable unit of the drum memory and each SECTOR contains ONE word. SECTORS 00 to 99 make up one TRACK.

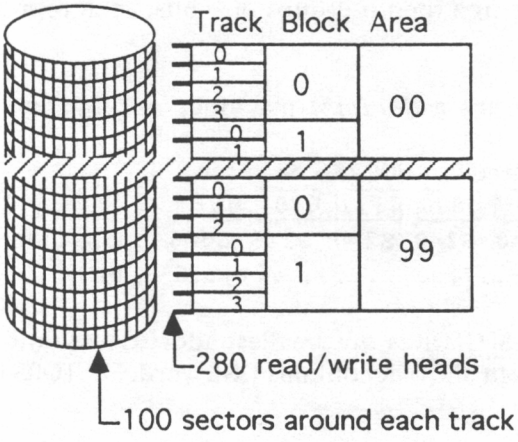
TRACK A TRACK occupies one revolution of the drum and each track is associated with one fixed read/write head. While there are 280 heads, the tracks are NOT numbered sequentially. Bailey developed a hierarchical addressing system to simplify the processing units. One to ten TRACKS (identified as 0 to 9) make up a BLOCK.

⌦ BLOCK This intermediate address covers a number of TRACKS. Up to ten BLOCKS (0 to 9) may appear in one AREA.

AREA This top level address range covers the whole drum memory. Between one and one hundred AREAS (00 to 99) could be created, though the 756 only used 00 to 09.



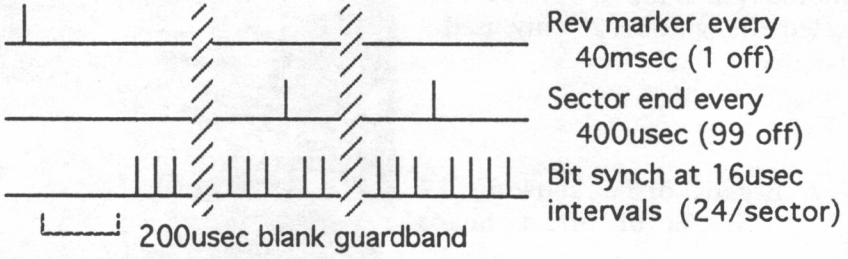
A Bryant drum showing 12 stacks of offset heads



Drum memory

In this example AREA 00 contains 2 blocks, AREA 00 BLOCK 0 contains 4 tracks, AREA 00 BLOCK 1 contains 2 tracks etc. Selecting AREA 00, BLOCK 1, TRACK 1 will give the 6th read/write head from the top.

So any AREA/BLOCK/TRACK value sent to the drum memory will result in one read/write head being chosen. The SECTOR requested has to be matched to the rotating position of the drum by using an extra 4 synchronisation tracks which give the drum start, the sector positions *and* the bit timing.



(the 4th synch track looks just like the 3rd, only delayed)

Drum synchronisation tracks

What the drum memory *actually* contains is:

Designated use	Words
Input data & computed results	1,000
Analog channel control	600
Utility words	800
High alarm limits	600
Low alarm limits	600
Variable high alarm limits	200
Variable low alarm limits	200
Data acquisition program	200
Periodic log program	400
Daily/weekly log program	200
Supervisory log program	200
Spare log program	200
Computer programs	10,000
Compiler programs	100
Compiler data	4,000
Alarm state	200
Magnetic drum synch	400
TOTAL ALLOCATED	19,900
SPARE	8,100
CAPACITY	28,000

(approximately equivalent to 70 Kbytes)

Even though these areas are of varying size the addressing scheme was arranged so that associated data could be used simultaneously.
eg (with AA Area, B Block, T Track, SS Sector)

	AA	B	T	SS
an input data value stored at	00	1	0	42
has its associated				
high alarm limit stored at	01	1	0	42
low alarm limit stored at	02	1	0	42
utility word stored at	03	1	0	42

As the sector gives the drum position all these values can be read, and checked, together!

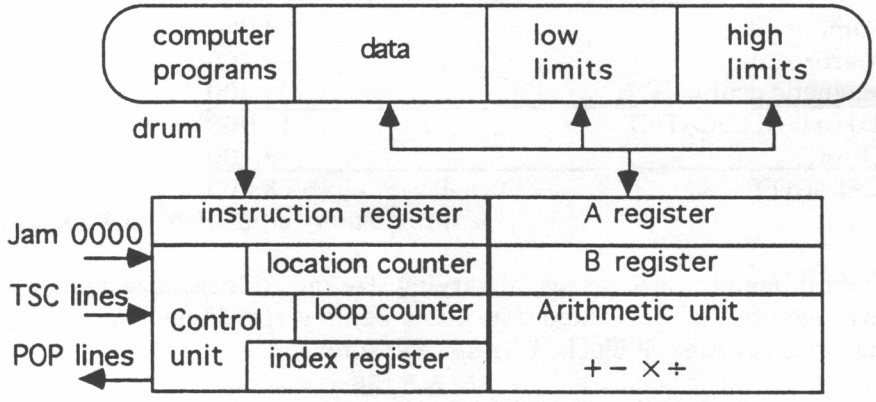
Also see DATASHEET 1

Computer

In power station operations *Data acquisition* is king and uses *Computer* as a slave, but I'll describe the computer first.

Though the systems operate in parallel, and a number are programmable, there is only one that can *calculate.*

The computer is intended to do arithmetic and some logical checks to fill entries destined for supervisor log printouts. The *Data acquisition* sub-system pulls on the computer's chain when work is ready to do (it causes a sort-of interrupt using its "POP" & the computer's "TSC" lines). It has 27 instructions, and 10 of these allow *interesting* addressing and the use of an index register.



Computer

The computer's control unit reads instructions from a predefined drum *Program area* into an *Instruction register.* The instructions are signed 6 digit numbers just like everything else in memory. The first two digits gives the operation code which leaves four digits to specify a

drum address. The *location counter* register holds the address of the next instruction and is four decimal digits as is the *index register.* The *loop counter* register is intended to match the "sectors" round the drum and has two digits.

The 756' address range of 0000 to 9999 is roughly equivalent to 24K bytes - not bad for 1962.

Some instructions for the Control unit are:

`TRU` *aaaa*	TRansfer Unconditionally to *aaaa*
`TOF` *aaaa*	Transfer on OverFlow to *aaaa*
`TAN` *aaaa*	Transfer to *aaaa* on A Negative
`SIR` *iiii*	Set Index Reg to *iiii*
`IRT` *aaaa*	Increment Index Reg & Transfer if > 0
SLC *cc*	Set Loop Counter to *cc*
`ICT` *aaaa*	Increment Loop Counter & Transfer if >0

The *arithmetic unit* operates on the *A register* or a double length (12 digit) register of A and B together with one argument from memory. Some more instructions are

`LDA` *m,aaaa*	Load A register from memory
`STA` *m,aaaa*	Store A register into memory
`ADD` *m,aaaa*	Add memory to register A
`SUB` *m,aaaa*	Subtract memory from register A
`MUP` *m,aaaa*	A × memory to give AB double register
`DIV` *m,aaaa*	AB ÷ memory with result to A and the remainder in B
`IAB`	Interchange A & B

For these commands *aaaa* represents a partial memory address and *m* is a required address modifier which may be[1]

---

[1] I'm cheating here - Bailey didn't use an assembler-like notation, just numbers with the instruction mnemonic as comment.

D	to address the Data area
H	to address the High limits
L	to address the Low limits
P	to address the Program area
*m*I	as above *and* add the Index register to the given address

The remaining commands include

`HLT`	Halt & wait for an external signal (TSC line)
`TSC` *nn,aa*	Test Signal Condition on TSC line *nn*, if it's on transfer to sector *aa*
`POP` *nn*	Put Out Pulse to line *nn*
`EXA`	EXecute the instruction in register A

Commands are normally taken one after another from sequential memory locations, and associated memory references (like ADD 0042) must wait for the drum to rotate to the required spot (like sector 42), then the drum has to get back to the location for the next instruction. Well, this might be a lot of waiting - some computers which used slow, rotating memory used an "optimising" procedure to put the "next" instruction in some place to minimise rotation wait time. So does the 756. If the minus sign is set in front of an instruction then the next instruction executed will come from the address of the **operand** +2. This is described more in the Programming section, along with variations for MUP, DIV and Indexing!

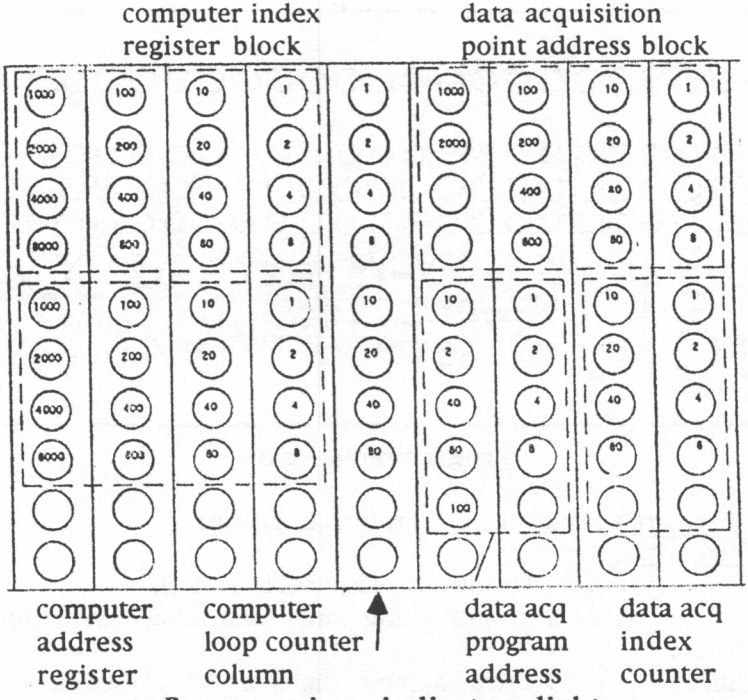
The computer can run at a maximum of 2500 instructions/sec so long as it does not refer to memory. That's not very useful and a more realistic *upper* speed is around 1000 instructions/sec (0.001 MIPS).

Also see DATASHEET 2

- Programming panel

The Programming panel *looks* like part of the computer, especially when it has its Flexowriter attached[2], but it is more than that. It is an independent subsystem that can change ANY part of the drum memory. It can load, edit or print computer program tracks, data acquisition channel words, log programs etc etc.

It *does* have some Computer specific buttons *and* a Computer display panel above it *and* Data acquisition specific buttons.



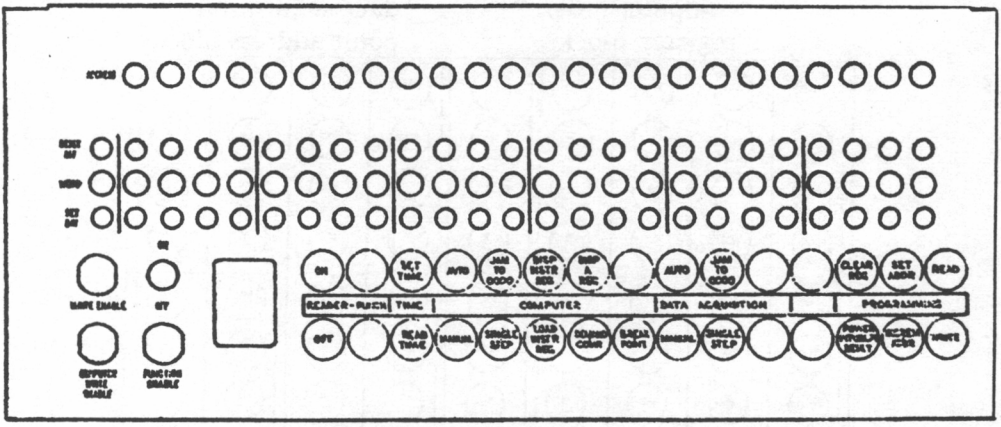
Programming indicator lights

---

[2] ACMS didn't get a Flexowriter or the other printing bits.

The Programming panel contains:

Address:	6 digit memory address in lights.
Reset Bit:	25 pushbuttons to clear bits in *Word*.
Word:	sign and 24 bcd digit bits.
Set bit:	25 pushbuttons to set bits in *Word*.
Write enable:	keylock switch to enable drum writing.
On/Off:	to turn on the panel lights.
Computer write enable:	keylock to write enable the computer drum memory area.
Function enable:	keylock to enable the panel.
00	Thumbwheel 00-14 used to request low-priority calculations.



Programming panel

Then there are the backlit pushbuttons in groups:

Reader-Punch group:

On:	enable Flexowriter (& disable panel).
Off:	enable programming panel (& disable Flexowriter).

Time group:

Set time:	from *Word* as hhmmss digits, *and* day of week from a thumbwheel in the cabinets as 1=Mon to 7=Sun.
Read time:	into *Word* as hhmmss.

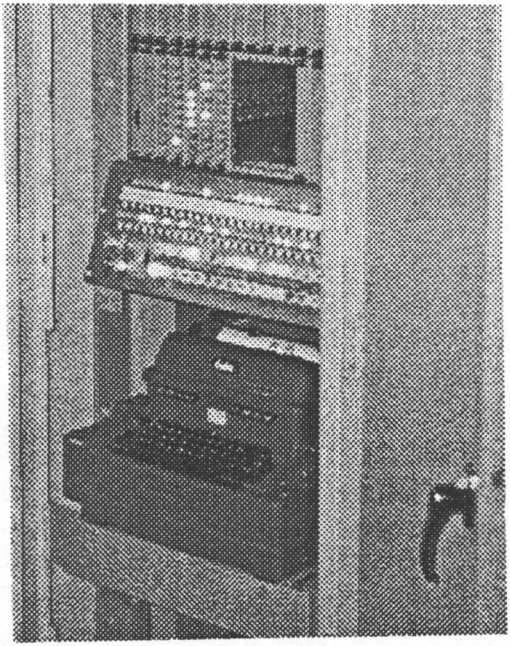
Computer group:

Auto:	set normal automatic operating mode.
Manual:	cancel *Auto* mode & enable Single step (etc?).

Jam to 0000: restart the program at 0000.
Single step: in manual mode execute next instruction.
Disp instr reg: put instruction register into *Word.*
Load inst reg: transfer *Word* to instruction register.
Disp A reg: put A register into *Word.*
Data acquisition group:
Auto: set normal automatic operating mode.
Manual: cancel *Auto* mode & enable Single step (etc?).
Jam to 0000: restart the program at 0000.
Single step: in manual mode execute next instruction.
Programming group:
Clear reg: zero *Word.*
Power interrupt reset: backlit to indicate a power interruption (the clock must be reset) press to clear.
Set addr: transfer *Word* to *Address.*
Increment addr: increment *Address* low 2 digits (sector #).
Read: transfer drum contents at *Address* into *Word.*
Write: transfer *Word* to the drum at *Address* .

The Programming panel uses the drum while other systems are operating by "stealing" revolutions from them. It "hides" 1 revolution in 40 for pushbutton and keyboard entry, but 1 in 6 for tape reader operations.

Also see DATASHEET 3



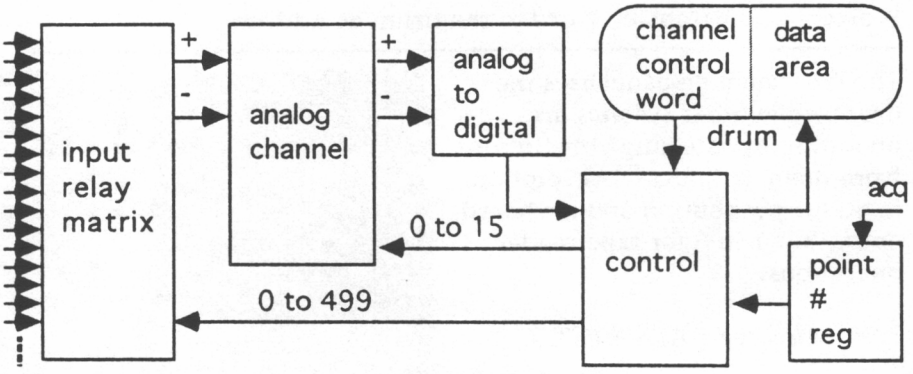
Programming panel and Friden Flexowriter

Data acquisition

- Analog

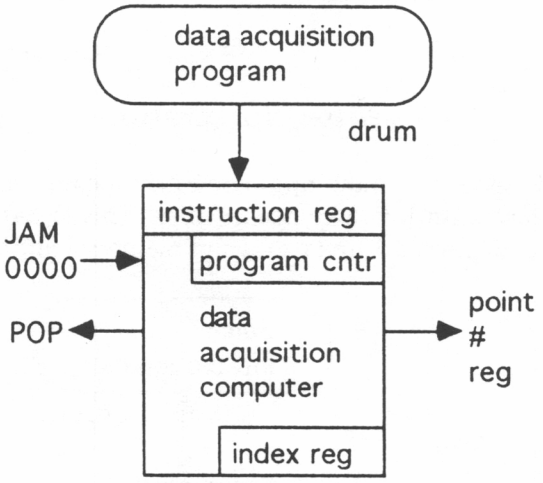
The Analog system is central to monitoring the 756's Power station. It is the source of quality and efficiency measurements required to manage the system and predict problems. There was a major investment in sensors for temperature, pressure, flow, levels and chemistry in nearly 500 points throughout the plant.

A complex double sub-system handles all this:
1) a hard-wired system to switch sensor input, do initial analog processing, convert analog levels to digital and store this on drum;
2) a programmable system which steers the data collector and requests processing by the computer.



Analog to Digital front-end

Control gets a four digit *point #* drum address and reads the data point's *channel control word* (CCW). This provides an input line number for the relay matrix (one of 500 signals) and an analog conditioning channel number (3 amplifiers and 30 channels to make anything into linear 0 to 4V). The *point #* is also the drum address to write the converted digital value (0 to 3999) into the *data* area. It does this 12 times a second and *point #* is incremented ...OR... altered by the data acquisition computer.



Data acquisition computer

The computer can set the next point to be collected by

`SPR` *pppp* set point address register to *pppp*

and successive points will be progressively stored away. The computer can track data collection by

`TPR` *pppp* halt until point reg = *pppp*

then it can redirect the collector. It uses this to sample some points **fast**: 51 points every 10 seconds for the console meters and fault compiler,
slower: 152 points every 45 seconds for the trend recorders and performance calculations,
slowest: the remaining 372 points in 90 seconds.

It uses the Index register to steer round its point list:

`SIR` *ii*	set index reg to *ii*
`IIR`	increment index reg
`TIR`	transfer to the sector # in index reg

and a command to instruct *other* systems:

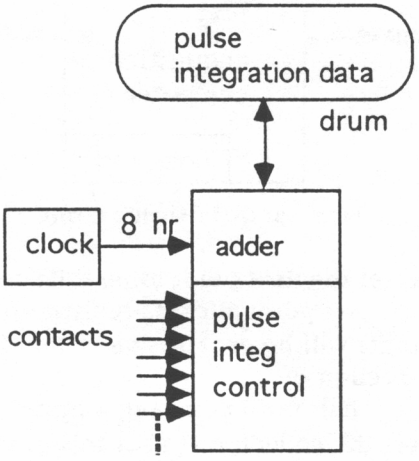
`POP` *p*	put out pulse on POP line *p* to start the fault compiler, restart the computer or

request an action by the computer.

Also see DATASHEET 4

- Pulse integration

A second whole section of data input comes from counting how often a collection of Power station switches go click. This is hard-wired logic that scans the switches and updates drum memory counters.

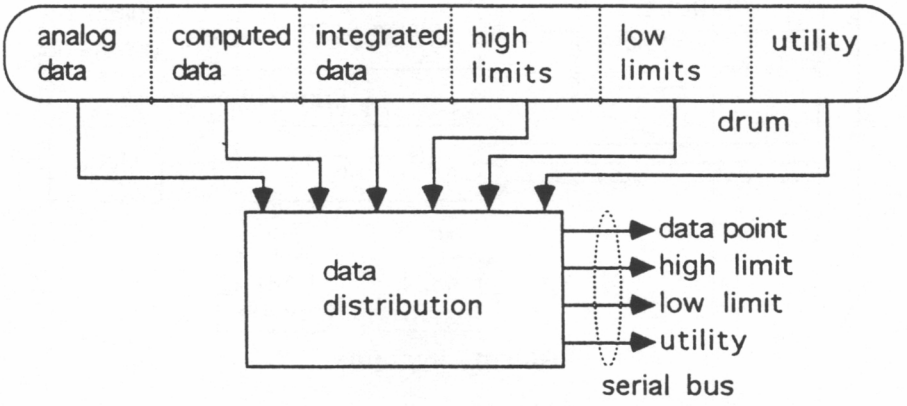


Pulse Integration

Contacts are scanned at 12 per second and for each closure 1 is added to the memory sector corresponding the switch. Up to 50 switch closures are counted in the first 50 locations of the data track. Each 8 hours these counts are transferred to total memories in the second 50 sectors of the track and the counts are cleared. These totals are used for regular log printouts.

Information distribution

The remaining systems need to read the data areas (& *not* write to them). To limit competition for the drum the Information distributor continuously spits entries from the data areas onto a shared bus at 2,500 points/sec.



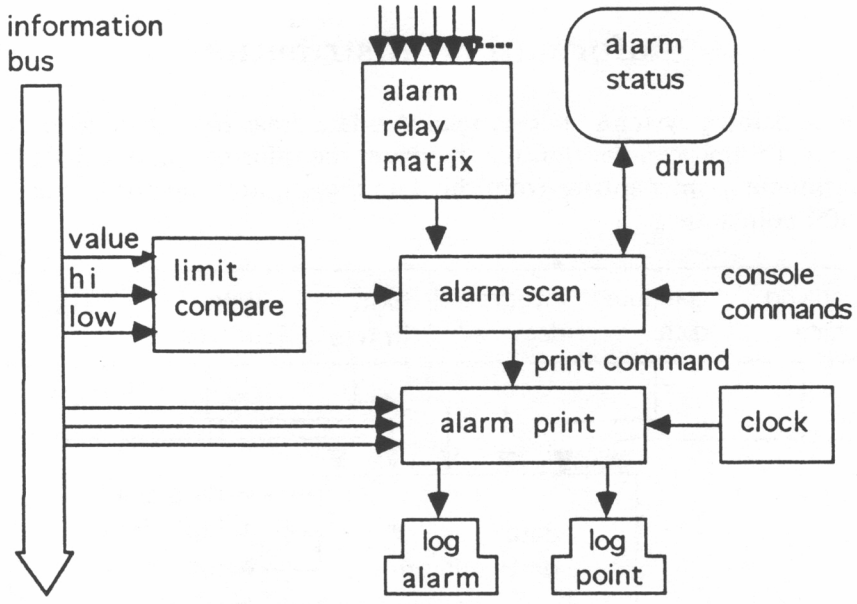
Data distribution

It cycles through 20 tracks (covering analog, computed & integrated data) with 100 points per track and simultaneously transfers value, limits and utility (including point number) onto the information bus.

Two types of system are attached to the bus: the alarm system checks the points at full speed while the slow printing systems casually take required values as needed.

Alarm

The Alarm system does three things: it **compares** point values to their stored limits and shouts if a problem is indicated; it **scans** the comparator's output, *and* alarm switches, against its memory record & screams if one changes; finally ... the **print** system does the actual shouting and screaming.



Alarm systems

The *comparer* checks limits at the speed of the information bus and provides high & low alarm status to the *scanner.*

The *scanner* reads alarm status in time with the *comparer's* output and if an alarm comes **on,** *or* goes **off,** an alarm printout is triggered.

Each sector in the drum's alarm track matches a data point value in the data area and is stored as one bit for high alarm on or off and a second bit for low alarm on or off. Successive data area tracks are mapped to subsequent bit pairs in alarm sectors. A final bit in each sector tracks up to 100 alarm switches. ie

bit:	12	11 10	9 8	7 6	5 4	3 2	1 0
alarm/track:	switch	lo 5 hi	lo 4 hi	lo 3 hi	lo 2 hi	lo 1 hi	lo 0 hi

Alarm printout can also be triggered by an alarm review request from an operator console, also, individual point values can be requested. These are printed in the same format but on a different printer.

The print system prints new alarms in RED and alarm end in BLACK as

```
hhmm
nnnn Q ±vvvv
```

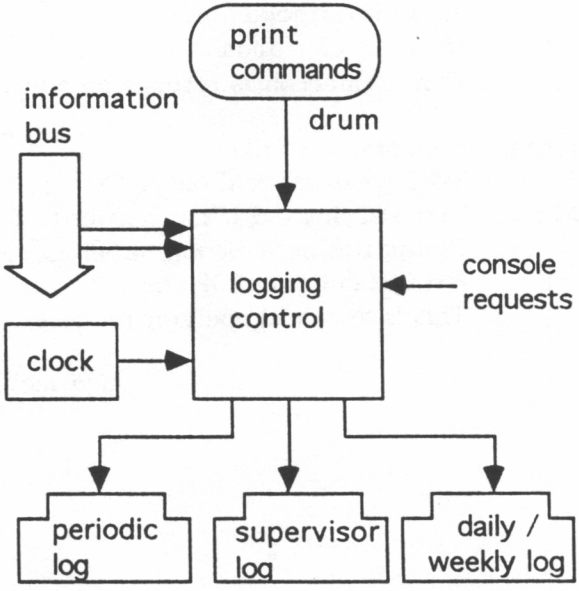
where *nnnn* is the point number, *Q* is H for high, L for low, C for closure and N for no-alarm-now, finally *vvvv* is the value.

New alarms also turn on a light & a siren (which the operator cancels).

Logging

- Periodic log

The 756 does a great deal of carefully formatted printing - up to 334 characters across pre-printed stationary as a periodic, one line, "report". It does not print characters, only numbers, but it has a whole programmable processor to do that.



Logging

The periodic log can be set to 15, 30 or 60 minute intervals *or* it can be produced at the press of an operator's button. The supervisor log consists of computer calculated heat loss measures & boiler efficiencies printed every 15 minutes, plus operational deviation measures every hour & summaries every day AND for a whole week.

The basic printing command is

`DS aaaa` Print data point *aaaa* formatted as *DS*

where

D is 1 to 6 to print a decimal point before the D^{th} high order digit, or 7 for no point,

S is 1 to 5 to print 1 to 5 high order digits.

eg for value = 543210 & DS = 24 then `5.432` is printed

Some other useful print commands include

`83`	Take a new line
`85 n`	Print *n* spaces
`87`	Select Red ribbon
`88`	Select Black ribbon
`91 aaaa`	Print number *aaaa* (usually an address)

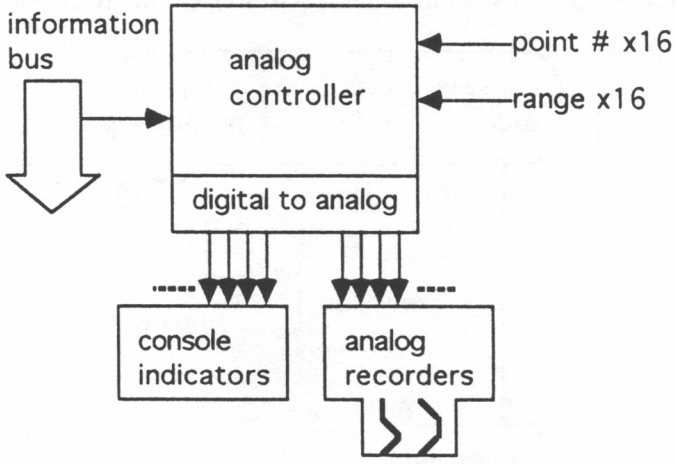
Along with some programming stuff like

`92 TSS`	Transfer to track *T*, sector *SS*
`93 C TSS`	Test TSC line *C* & if set transfer to *TSS*
	C indicates periodic log, supervisor log etc
`94 P`	Put Out Pulse on POP line *P*
	This is to cancel operator requests.

Also see DATASHEET 5

- Analog strip recorders etc

The other form of logging involves continuous output of analog values to strip recorders and to indicators on the operator's consoles.



Analog trend & displays

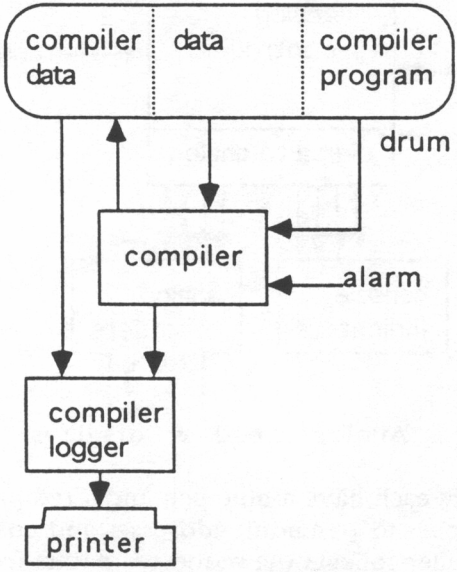
Eight strip plotters each have a blue pen and a red pen with thumbwheel switches to give point addresses and pushbutton scaling range. The controller collects the requested points from the bus and converts them BACK into analog from whence they started, and plots at a mind-boggling speed of 2" per hour.

It also has a "wired in list" of 24 points to collect and convert for the console's analog meter displays.

ETC

- Fault compiler

The fault compiler is a nifty gadget that normally skulks about in the background hoarding some critical point values. If the alarm system gets alarmed then the fault compiler keeps its current collection, makes another and prints out a before-and-after the event log.



Fault compiler

The compiler continuously copies 40 points at a leisurely 1 every 10 seconds. It collects a full set of 30 samples in 5 minutes and lights "compiler full" on the operator's console. On a fault (or at operator request) these are retained and a *further* 60 samples are taken in 10 minutes. Then pre & post event values are printed.

The compiler "program" is just a list of BTSS data area addresses.

- Clock

The clock triggers a variety of regular events at intervals ranging from 15 minutes to 1 week. The time can be set at the programming panel as day-of-week, hour, minute and second.

certified Y2K safe [3]

Time is also supplied to various printing systems.

- Timer

There are 28 timers stored on a track in the data area. The first 14 (in sectors 02 to 28) are incremented every second and at 999999 they continue to 0 and onward forever. The second set of 14 (in sectors 30 to 56) are decremented each second and STOP at 0.

The computer uses these in association with cunningly set alarm limits to check correct operation.

---

[3] A curious preoccupation at the time of writing.

Programming the 756

Calculations are actually done with integers ranging from -999,999 to 0 and up to 999,999. While this is a lot of numbers it doesn't immediately help if you need to calculate acidity of the steam condensate waste water in the range 1.00000 to 7.99999. What you *can* do is to use numbers like 100000 to 799999 and "pretend" that there is a decimal point after the first digit. The print system supports this by having formats which plug in a decimal point on demand.

That's a good start, but calculations always seem to want to use numbers with different assumed decimal point positions. To keep track of this you could build floating-point arithmetic hardware, *or* do as the Bailey programmers did and keep track of the decimal point scaling. Their program documentation notes this scale as "Q" where

Q = 0 gives .799999 (0 places before the DP)
Q = 1 gives 7.99999
...
Q = 6 gives 799999. (6 places before the DP)

eg solve [4] $R=\frac{(M+N)0.9+S}{100T}$

First allocate some locations for the numbers in the Data area:

address	value	Q	comment
000400	600000	1	M = 6
000401	020000	3	N = 20
000402	043000	2	S = 4.3
000403	750000	2	T = 75
000404	x	1	location for the result

Then the program in the Program area:

address	command	Q	comment	register A
080400	LDA 0400	1	load M into A	600000
01	SR2	3	shift right to fix Q	006000
02	ADD D,0401	3	(M+N)	026000

[4] From Bailey Maintenance Manual 3.5/20

```
03    MUP P,0410   3  (M+N) x.9                  023400
04    SL1          2  shift left to fix Q        234000
05    ADD D,0402   2  (M+N).9+S                  277000
06    SR3          3  ((M+N).9+S)/100            000277
07    DIV D,0403   1  ((M+N).9+S)/100T           003693
08    STA D,0404   1  store A to result
09    HLT              & halt.
10    900000       0  constant 0.9
```

Note that the decimal digit shifts at locations 01 and 04 are needed to match the Qs (decimal position) before addition. At 06 a shift is used to divide by 100 *and* to adjust Q. For multiply you add Qs and for divide you subtract the divisor's Q. Phew, now you know what floating point hardware does!

Another example to total the values in data area 0 to 10:

```
address   command        Q  comment
080000    LDA D,0000        load the 1st value in A
    01    SIR 9990          set the index register to -10
    02    ADD ID,0011       add at (11-10=1), 2, 3...
    03    IRT 0002          incr index reg & transfer to 02
    04    HLT               until index reg = 0
```

The instruction at 01 loads the 4 digit index register with -10 as a *10's complement*. The indexed add at 02 uses an effective address which is the sum of the given address (11) and the value of the index register which goes from -10 to -1. The IRT instruction at 03 normally transfers back to the add, but when the index register increments from 9999 to 0000 it goes on to the next instruction. (The value of Q is constant in this example)

A really useful programming operation is the subroutine call. The Bailey manuals describe a number of numerical routines and a standard calling method:

```
address   command          Q  comment
          LDA argument        load argument value in A
          IAB                 & put it in register B
```

```
        LDA  m          load a return instruction in A
        TRU  subr       & transfer to the subroutine
m       TRU  n          the return instruction
n       ...             next instruction to do
```

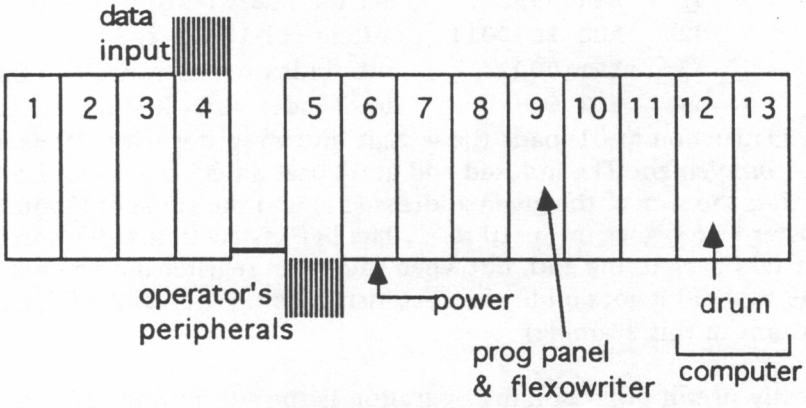
Then the subroutine does something like:

```
subr    IAB             save return instruction in B
        ...             make a value to return in A
        IAB             value to B & return instr to A
        EXA             execute the instruction in A
```

which gets you back to *n* in the calling program with a calculated result in register B.

Hardware

The Bailey Meter 756 consists of 13 racks of electronics each 2' wide by 2' deep by 7' high made from $^{1}/_{8}$" steel.



Bailey 756 cabinets

Sub-systems are not neatly parcelled in separate cabinets but form interlocking regions of functionality:

switching relays	1,2,3	prog panel & flexowriter	9
analog processing	1,2,3,4	prog control	10
pulse integration	4	alarm limit entry	10
alarm scan	4,7	limit compare	10
alarm printer	5,6	fault compiler	11
data acquisition	5,6	drum read-write	11
logging	6,7,8	info distribution	11
console	6,7,8,9,10	drum timing	11
clock	7	computer	12,13
programming printer	8	Bryant 7514 drum	12
fault printer	9,10		

Cabinet distribution of functions

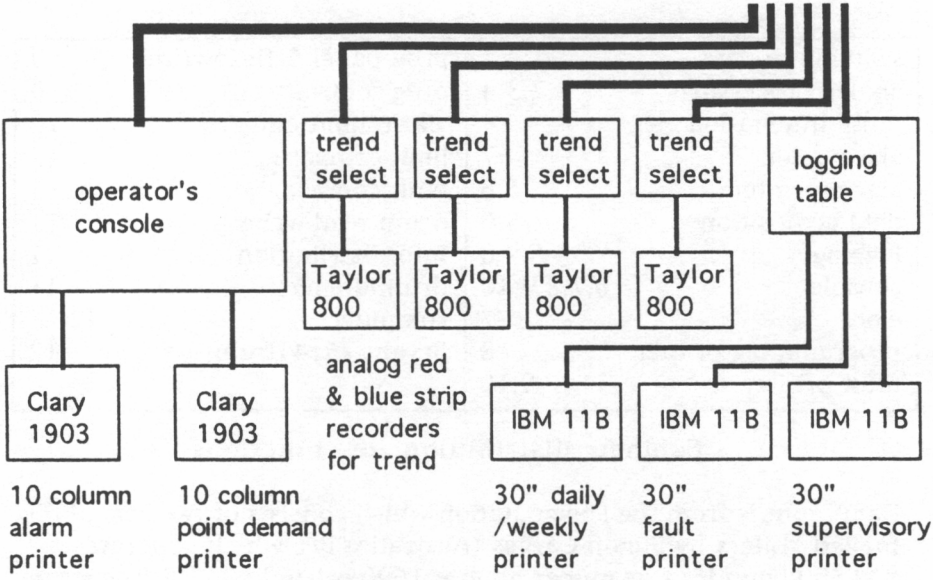
Data input is from the Power station which consists of two sets of oil fuelled boilers by Simon-Carves (Australia) P/L which generate 822,500 pounds of steam per hour at 1600 psi and 1005°F. The steam drives C.A.Parsons & Co Ltd 120MW tandem compound turbines. Each is monitored by

- 195 thermocouple temperature sensors, (connected to "cold" junction boxes held at 150°F in cabinet 1 & 2),
- 20 resistance temperature sensors,
- 49 pressure, flow & level sensors,
- 9 chemical sensors,
- 12 electrical sensors,
- 5 time of operation switches,
- 84 alarm switches,
- plus 2 system check.

ie 376 sensors per unit.

The Operator's peripherals are connected by 50' cables. There were two identical operator stations, one for each boiler-turbine-generator set. [5]

[5] ACMS did not get any of this.



Operator's equipment

In addition the 756 was supplied with
- a Tektronix 535A oscilloscope,
- resistance box,
- potentiometer,
- multimeter,
- wheatstone bridge &
- Bailey "cheater cords".

BAILEY 721

The ACMS obtained the three cabinet Bailey 721 system along with the 756. This *IS* the control system for the power station.

It is not part of the 756.

It is not a computer (in any modern sense).

It is not digital.

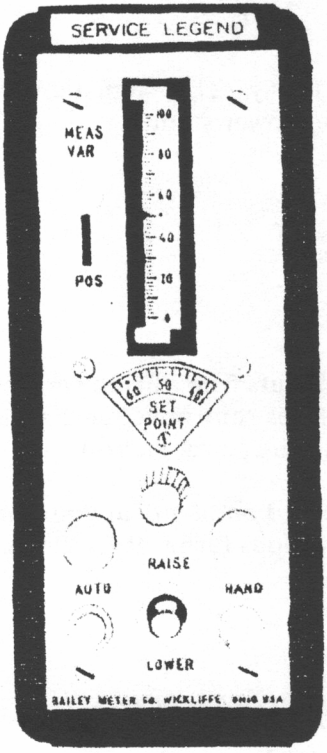
Yep. It takes some of the same analog inputs from the power station, processes them analogically and generates control voltages which are applied to motors, pumps, valves etc in the power station.

The 721 uses modules which are all based on identical operational-amplifiers (with gain >10,000) using various forms of feedback to provide functions including:

[%]	proportion
[Σ]	sum of two inputs
[Δ]	difference of two inputs
[$^{d}/_{dt}$]	derivative (or rate of change)
[∫]	integral or sum of one input
[$^{\Sigma n}/_{N}$]	average of input
[%=f(x)]	calculated value
[X=f(t)]	delay passing input value
[>]	limit high, [<] low, [><] both
[<]	select lower with two input
[S]	set point
[+-]	bias

Which are sometimes combined in one unit like [%+∫]. These modules are plugged together on their front panels to form complete closed-loop control sub-systems. Each of these sub-systems is controlled by a

panel on the operator's 721 console:



An analog meter showing either the VARiable being controlled, or the POSition of the controller

A dial and knob to give the SET POINT for the sub-system

A red light and pushbutton to set AUTOmatic loop operation

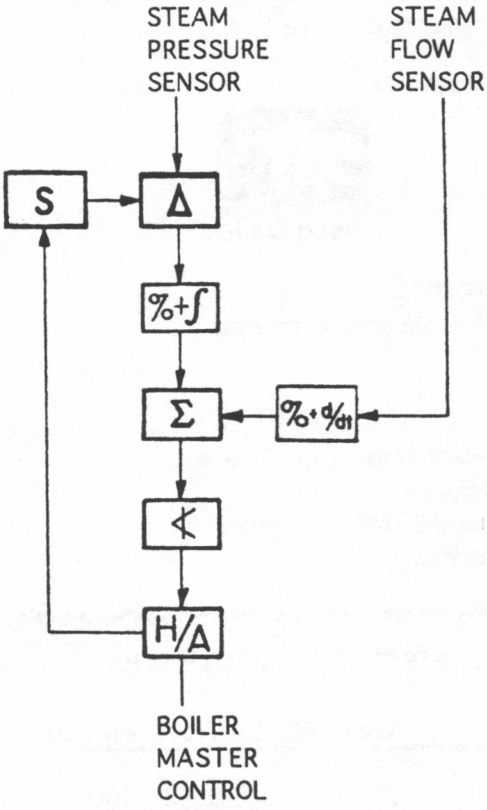
A white light and pushbutton to set HAND or manual operation

A three position switch to RAISE, leave or LOWER the controlled element

Control loops were built for

- *Boiler Master,*
- *Fuel oil flow,*
- *Air flow,*
- *Feedwater flow,*
- *Steam superheat temperature* and
- *Steam reheat temperature.*

As an example, the first system control sets how hard the boiler should be working:



Difference between Set point and measured Pressure,

Take a Portion + Integrated trend,

Sum that with a Portion + Rate of change of Steam flow,

Limit to a maximum value,

Choose between Hand and Automatic control.

Postscript

The Bailey Meter company still exists as Elsag Bailey, and is still doing the same thing. They can be visited at

http://www.ebpa.com

In early 1999 this proclaimed:

Elsag Bailey Process Automation, NV,
is a leading provider of automation solutions to process industries worldwide.

News:

July 1998 *Louisiana Power Plant Automation Contract Awarded to Elsag Bailey.*

August 1998 *Elsag Bailey Awarded $9 Million Contract for German Power Station.*

Datasheet 1 - Drum Memory Addresses

CONTENTS	Area	Block	Track	#words
---Data---				
analog in unit 1	00	1	0 - 2	300
computed results unit 1	00	1	5	100
alarm contact addresses unit 1	00	1	6	100
time unit 1	00	1	7	100
integrated data units 1	00	1	8	100
analog in unit 2	00	2	0 - 2	300
computed results unit 2	00	2	4	100
alarm contact addresses unit 2	00	2	6	100
time unit 2	00	2	7	100
integration data units 2	00	2	8	100

--High limits--				
fixed limits for analog data unit 1	01	1	0 - 2	300
limits for comp'd results unit 1	01	1	4 - 5	200
fixed limits for analog data unit 2	01	2	0 - 2	300
limits for comp'd results unit 2	01	2	4 - 5	200
--Low limits--				
fixed limits for analog data unit 1	02	1	0 - 2	300
limits for comp'd results unit 1	02	1	4 - 5	200
fixed limits for analog data unit 2	02	2	0 - 2	300
limits for comp'd results unit 2	02	2	4 - 5	200
--Channel control words--				
CCW unit 1	03	1	0 - 2	300
CCW unit 4	03	2	0 - 2	300
Data acquisition program	04	0	0 - 1	200
Compiler program	05	0	0	100
--Compiler data--				
compiled data unit 1	06	1	0 - 9	1000
compiled data unit 1	06	2	0 - 9	1000
compiled data unit 2	06	3	0 - 9	1000
compiled data unit 2	06	4	0 - 9	1000
--Printer programs--				
periodic log unit 1	07	0	0 - 1	200
periodic log unit 2	07	0	2 - 3	200
supervisory log	07	0	4 - 5	200
daily/weekly log	07	0	6 - 7	200
Computer programs	08	0 - 9	0 - 9	10,000
--Utility words--				
analog UW unit 1	09	1	0 - 2	300
computed results UW unit 1	09	1	4 -5	200
analog UW unit 2	09	2	0 - 2	300
computed results UW unit 2	09	2	4 - 5	200

Datasheet 2 - Computer Programming Instructions

Command format is either OOAAAA where OO is operation code and AAAA is memory address (or operation specific digits), or MOAAAA where M is a modifier for what address is used. ie

0	data area	4	data area + Index
1	high limit	5	high limit + Index
2	low limit	6	low limit + Index
3	program area	7	program area + Index.

and X indicates "digit doesn't matter"

M0XXXX	HLT	Halt until external signal (or NOP in manual mode)
M1AAAA	LDA	Load A reg
M2AAAA	STA	Store A reg
M3AAAA	STR	Store index reg
M4AAAA	STC	Store loop counter
M5AAAA	ADD	Add memory to A (2 sector times)
M6AAAA	SUB	Subtract memory from A
M7AAAA	MUP	A × memory to give AB (A high, AB same sign) avg 10ms
M8AAAA	DIV	AB ÷ memory, quotient to A & remainder in B
M9AAAA	TAE	Test A Equal to memory, if equal set overflow
80NNXX	SRX	Shift AB NN digits right, max 6 (2 sector times)
81NNXX	SLX	Shift AB NN digits left, max 6 (2 sector times)
82XXXX	IAB	Interchange A & B
83TTSS	TSC	Test TSC line TT (00 to 09), if on transfer to sector SS
84PPXX	POP	Put Out Pulse to line PP (00 to 09) for 1 sector time
85AAAA	TRU	Transfer Unconditionally
86AAAA	TOF	Transfer on Overflow & clear overflow
87AAAA	TAN	Transfer on A Negative
88AAAA	IRT	Increment Index Reg & Transfer if not 0
89AAAA	ICT	Increment Loop Counter & Transfer if not 0
90NNNN	SIR	Set Index Reg from NNNN
91XXNN	SLC	Set Loop Counter from NN
92XXXX	CLA	Clear A
93XXXX	CLB	Clear B
94XXXX	NOP	
95FTXX	CPR	Copy Register according to FT =

12: A to B 13: A to Index 14: A to Loop
31: Index to A 41: Loop to A

96XXXX	EXA	Execute instruction in A register
97XXXX	unused	
98XXXX	unused	

99XXXX unused
+commands take the next sequential location next
- commands can be used to "optimise" execution time: they take the next instr from the argument's sector addr +2 (except MUP & DIV where the next address is the argument sector address + 50). In both cases the addition wraps round the sector digits without affecting the track digit. Also, if the address was modified by indexing it is the *unindexed* address which generates the next location.

Instruction time is 1 sector (0.40 msec) if there is no memory reference, except ADD, SRX and SLX which take 2 sector times. With a memory reference 1 or 2 drum revolutions must be added (ie 40msec to 80msec) unless optimised (-code) sequencing is used. MUP and DIV require 50 to 150 sector times (20 to 60 msec). Also, index modification takes 2 extra sector times.

Datasheet 3 - Programmer's Panel Operation

Update system program etc through program panel:
a) Function enable on,
b) clear reg,
c) set address bits,
d) press "set address",
e) clear reg,
f) set program bits,
g) Write enable on,
h) write.

Print memory:
a) turn Flexowriter on,
b) align paper,
c) Function enable on,
d) Program station on,
e) type the 6 digit start address then P. All data from starting address to sector 99 will print.

Read paper tape into memory:
a) insert prepared paper tape with first holes being read,
b) Function enable on & write enable on,
c) Flexowriter on,
d) Program station on,
e) type the six digit starting address (if it's not prepunched),
f) type R. Data will be written to memory from the start address until sector 99 is written.

Paper tape format:
nnnnnn<CR/LF>*nnnnnn*<CR/LF>...$

Datasheet 4 - Data Acquisition Programming Instructions

The data acquisition program selects the point address on the drum and the analog processor reads the associated Channel Control Word (CCW) to obtain a point value. The CCW has format ANNRRR where
A selects the amplifier as
1 for Thermocouple type CC temperature sensor
2 for Thermocouple type CA temperature sensor
3 for gain 100
NN selects the correcting analog channel as 00 to 29 (see below)
RRR is the address in the relay switching matrix.

Channel	amplifier	divider/lineariser	transfer
0-13		*spare*	
14	CC T/C	0-600°F CC Thermocouple	×10
15	×100	0-600°F 10 ohm RTD	"
16-19	"	10:0.5, 10:0.6, 10:0.8, 10:1	"
20-26	"	10:1.4, :1.5, :2, :2.5, :3, :3.5, :4	×1
27	"	0-1200°F 10 ohm RTD	"
28	CA T/C	0-1200°F CA Thermocouple	"
29	×100	10:1.2	"

The data acquisition computer commands are formatted as XOAAAA where X is not used, O is operation and AAAA depends on the command. The sign is not used.

```
X0XXXX NOP
X1PPPP SPR    set point address register to PPPP
X2XXXX        nu
X3XXII SIR    set index reg to II
X4XXXX IIR    increment index reg
X5PPPP TPR    halt until point reg = PPPP
X6XXXX TIR    transfer to sector in index reg
```

`X7XXXX` nu

`X8PXXX POP` put out pulse to POP line P ie:
0 to start fault compiler,
1 to set point address to unit#1 console TWS,
2 to set point address to unit#2 console TWS,
3 JAM 0000 (restart) to computer,
4 computer action via TSC 20

Datasheet 5 - Print programming

The print processor has three instruction formats: -XXAAAA (X means doesn't matter) for unformatted print, DSAAAA with DS as 01 to 79 for formatted print and OOAAAA where OO is an operation code.

`-XXAAAA` Print data point AAAA unformatted, signed 6 digits

`00XXXX` Halt

`DSAAAA` Print data point AAAA formatted as DS (01 to 79):
D=1 to 6 dec point before digit 123456, D=7 no "."
S=1 to 5 print 1 to 5 high order digits, if value=543210 then
S=6 print 432, S=7 print 32, S=8 print 54:32 ie time,
S=9 print 4321 ?, S=0 prints 543210 ?? *(manuals are not consistant for S=9 & 0 and do not match the given range)*

`81XXXX` Lower case

`83XXXX` Print CR/LF

`85NXXX` Space N times

`86NXXX` Tab N times

`87XXXX` Red ribbon

`88XXXX` Black ribbon

`90XXXX` NOP

`91AAAA` Print address AAAA

`92XTSS` Transfer to track T, sector SS

`93CTSS` Test TSC line C & if set transfer to TSS where C=

1 unit 1 periodic log,	6 unit 1 demand print,
2 unit 2 periodic log,	7 8 hr log,
3 supvisor log,	8 unit 2 demand print,
4 daily log,	9 1 hr log,
5 weekly log,	0 NOP

94PXXX	Put Out Pulse on POP line P:	
	1 reset unit 1 periodic req,	5 reset weekly request,
	2 reset unit 2 periodic req,	7 reset 8 hr request,
	3 reset supervisor request,	9 reset 1 hr request
	4 reset daily request,	
95PXXX	Start printer P:	
	1 unit 1 per log printer,	3 supervisor log printer,
	2 unit 2 per log printer,	4 daily/weekly log printer
96XXXX	Stop all printers	

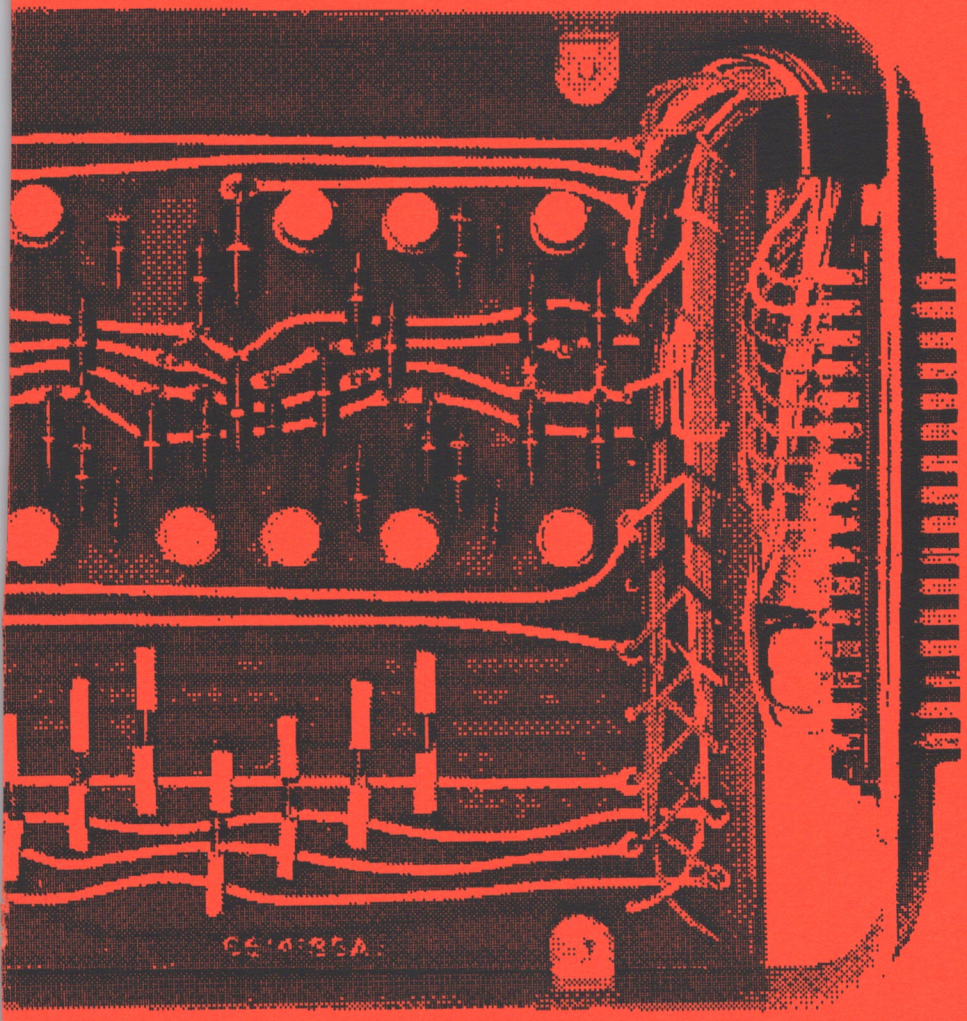
(other codes are not used)

Datasheet 6 - BAILEY Documentation

These are abbreviated titles. eg the full title for the first item is
B700 SYSTEM OPERATING SPECIFICATION. BAILEY 756 COMPUTING SYSTEM FOR THE ELECTRICITY TRUST OF SOUTH AUSTRALIA TORRENS ISLAND POWER STATION, JOB NO. 654J.

Operating specification BAILEY 756
BAILEY 756 Operating guide
BAILEY 756 Information system units 3&4
BAILEY 756 Instructions
BAILEY 756 Instructions units 1&2 vol. 5 of 5
Vendor data & technical manual BAILEY 756 vol. 2, 3 of 5
Instrumentation manual units 1&2 Section A Maintenance 756 & 721
BAILEY 756 maintenance schedule
BAILEY 756 maintenance manual units 3&4 (inc computer operations)
BAILEY 756 units 1&2 vol.2 records
Program documents vol. ?, 4 of 5
Engineering report
Wire list units 1&2

BAILEY 721 general description vol. 2
Instruction manual BAILEY 721 systems (boiler controls) vol. 1 & 2
BAILEY 721 Instructions, 3 volumes

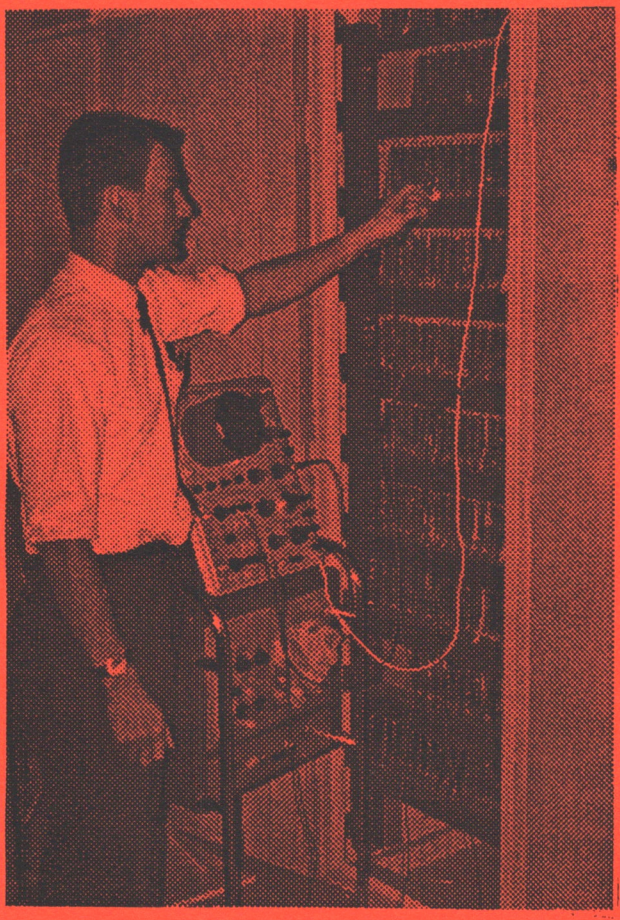


Other publications by the
Australian Computer Museum Society

CSIRAC - Australia's First Computer

The University of Manchester's BABY - the first modern computer

Back: ***ETSA maintenance engineer***



If it's **1962**
&
the transistors are
germanium
&
the memory is a
drum
&
there are **6**
processors
&
they are all
different
&
it weighs **8** tons

IT'S A
BAILEY 756
(even without its
Power Station)

ISBN 0 9585678 1 6

AUSTRALIAN COMPUTER MUSEUM SOCIETY