SOUTH AUSTRALIAN

MICROPROCESSOR GROUP

P.O. BOX 113,
PLYMPTON, S.A. 5038
TEL. 278 7288

INC.

Meetings held at
THEBARTON HIGH SCHOOL
ASHLEY ST., THEBARTON

NEWSLETTER

Vol. 1. No. 2.
June/July 1979 Newsletter. *Mail Box*, ...P.O. BOX 113, PLYMPTON S.A.

It is quite obvious from the number of systems and members present at the April meeting, that B.Y.O.S. nights are the most popular. I think that we shall need a larger room for the next systems night.

The May meeting was also well attended, and 10 new members joined our group, it seems there are a quite a number of enthusiasts and the interest seems to be growing, rapidly.

The June meeting will be centred around a visit from COMPUTERLAND to show us the APPLE system. I know there are a number of members who, as yet have not purchased or built up a system. This indeed, will be an ideal opportunity, to see a fully built up system with all the bugs ironed out. I understand the colour graphics on the Apple is something quite remarkable, with very high resolution.

Bob Stunell our Secretary/Treasurer has taken leave of absence to visit the U.K. I understand he will be looking at computer usage within the education system and as used by students. Hopefully he will return in time for the September meeting, so Bon Voyage Bob.

While Bob is away John Moffatt will be acting Secretary, so anyone wishing to join the group may see John at the meetings or write to the P.O. Box as above.

At this stage I would like to give a note of thanks to Mike Sunners who offered to construct the keyboard kit for the club 2650 system, and made a very neat job of putting it together, thanks again Mike.

Just after the last newsletter was released, we recieved an application for membership, from the T.R.S. 80 users group. We would like to take this opportunity to welcome them to our group. The following details will enable any interested people to contact them. Contact Mr. G.F. Stevenson at 34 Sturt St. Adelaide 5000 Phone 250 0178.

Lastly we have advised E.T.I. and E.A. of our group and the correct info regarding our current venue for meetings etc.

YOUR COMMITTEE FOR 1979

CHAIRMAN: Eric Clarke		Work Phone 278-7288
SECRETARY/TREASURER:	Bob Stunell	Work Phone 352-5811
ACTING SEC/TREASURER:	John Moffatt	

COMMITTEE MEMBERS:

Tony Beresford
Bob Daniells
Howie Harvey
Rick Matthews

FUTURE MEETINGS JUNE/JULY

As mentioned on the cover June will be a visit from Computerland. July will be a talk on the 6800 and 6809 by Rick Matthews. We would welcome any 6800 systems for those who would like to see the finer points of 6800 registers etc.

INCORPORATION OF THE GROUP

We have finally received the certificate of Incorporation, many thanks to those involved in the long process of this project.

6800 USERS GROUP.

There are a number of people in the S.A.M.G. who have 6800 microprocessor gear. They wish to advise anyone who is interested in coming along may contact: Mr Dave Jones 258-2494 or Mr Eric Clarke 278-7288.
Meetings are not held very regularly but if the demand requires I am sure this will alter.

CHANGE OF ADDRESS

It has come to our notice that some members are not receiving the newsletter we have also received a couple of newsletters back. Under the new postage rates which we have arranged, if we have mail returned to us because of wrong addresses then we have to pay again for the return postage.
It would be a great help to us if you notify any of the committee of any change of address etc.

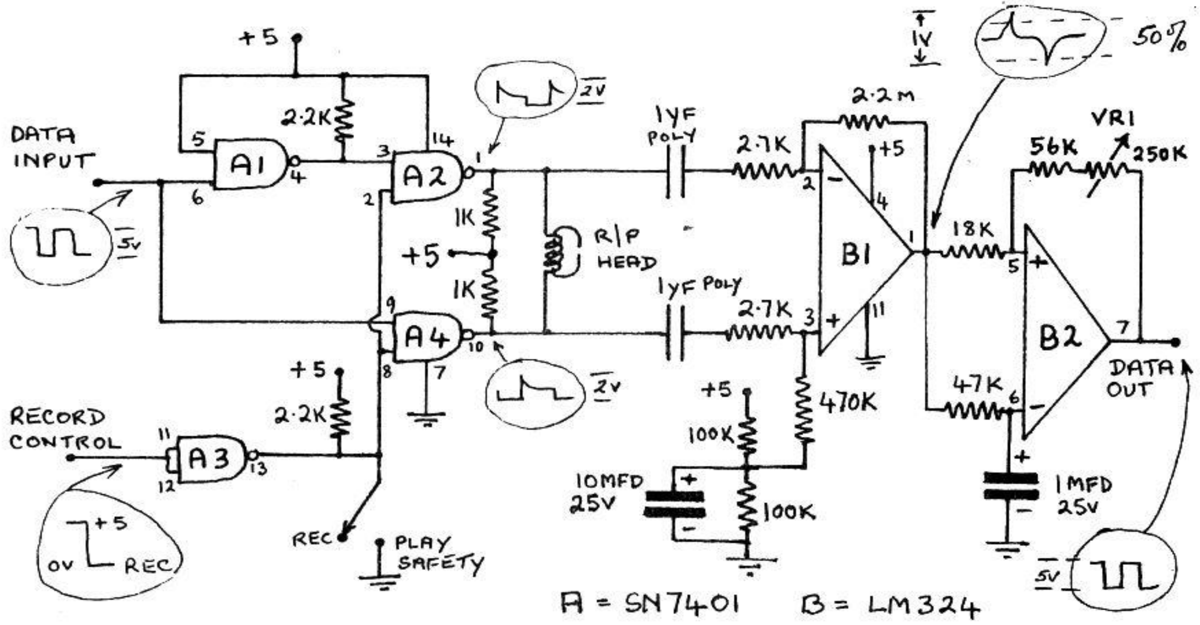
FOR SALE.

DG640 Memory Mapped VDU Kit.	$120
KT9500 (2650) Microprocessor Kit	$100

Kits have never been used.

Contact Mr. D. Oakes Phone 250-0178 After Hours.

N.R.Z. TAPE INTERFACE MODIFICATIONS BY Eric Clarke.



An article in April 1977 Electronics Australia by Ian Croser provided me with the idea of using NRZ recording instead of kanzas city which uses audio shift keying. My main reason for picking NRZ was the small component count and high data rate capability. Although my serial data interface will not run faster than 1200 Baud I have recorded 2400 Hz square waves quite nicely.

I have made a number of changes from the original circuit the main one being to cure the data errors due to D.C. drifting in the B1 differential amplifier. The drift was seriously upsetting the operation of the threshold detector B2. This was cured by adding the 47K and 1MFD to pin 6 of B2 thereby allowing both inputs of B2 to follow D.C. wise the output of B1.

Also a single 5v supply replaces the triple supply design and using a 7401 allows a record safety switch to be added to the R/P control input with no extra chip count. In fact with a chip count of three one could have a 2 channel or 2 deck system. I found the output from B2 will drive a TTL load but I would suggest using a 4041 or 4049 Cmos TTL driver for nice clean edges on your data train.

The only adjustment required to set up this cassette interface is VR 1 which should be adjusted for reliable data recovery. This means that the thresholds of B2 should be set at 50% of the peak pulse levels from B1.

Layout is not critical I constructed the prototype on veroboard and its still working that way. If you do not require remote control of the R/P gating just leave out the connection from pin 13 of A3 but still leaving the 2.2K on the switch. The switch will now operate as a R/P on off control.

As a final tip a 25mfd25vw Electro and a .1 Ceramic capacitor would go nicely across the supply lines on the board.

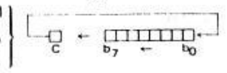
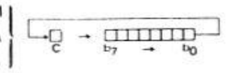
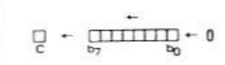
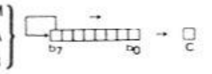
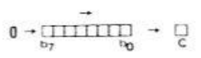
ACCUMULATOR AND MEMORY		ADDRESSING MODES															BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG.					
		IMMED			DIRECT			INDEX			EXTND			INHER			(All register labels refer to contents)	5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	●	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	●	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	●	●	↕	↕	R	●
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	●	●	↕	↕	R	●
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	●	●	↕	↕	R	●
Clear	CLR							6F	7	2	7F	6	3				00 → M	●	●	R	S	R	R
	CLRA													4F	2	1	00 → A	●	●	R	S	R	R
	CLRB													5F	2	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	●	●	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				$\overline{M}$ → M	●	●	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$ → A	●	●	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$ → B	●	●	↕	↕	R	S
Complement, 2's	NEG							60	7	2	70	6	3				00 − M → M	●	●	↕	↕	①	②
(Negate)	NEGA													40	2	1	00 − A → A	●	●	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	●	●	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	●	●	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	●	●	↕	↕	④	●
	DECA													4A	2	1	A − 1 → A	●	●	↕	↕	④	●
	DECB													5A	2	1	B − 1 → B	●	●	↕	↕	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A ⊕ M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B ⊕ M → B	●	●	↕	↕	R	●
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	●	●	↕	↕	⑤	●
	INCA													4C	2	1	A + 1 → A	●	●	↕	↕	⑤	●
	INCB													5C	2	1	B + 1 → B	●	●	↕	↕	⑤	●
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	●	●	↕	↕	R	●
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	●	●	●	●	●	●
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	●	●	●	●	●	●
Rotate Left	ROL							69	7	2	79	6	3				M 	●	●	↕	↕	⑥	↕
	ROLA													49	2	1	A	●	●	↕	↕	⑥	↕
	ROLB													59	2	1	B	●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M 	●	●	↕	↕	⑥	↕
	RORA													46	2	1	A	●	●	↕	↕	⑥	↕
	RORB													56	2	1	B	●	●	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M 	●	●	↕	↕	⑥	↕
	ASLA													48	2	1	A	●	●	↕	↕	⑥	↕
	ASLB													58	2	1	B	●	●	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M 	●	●	↕	↕	⑥	↕
	ASRA													47	2	1	A	●	●	↕	↕	⑥	↕
	ASRB													57	2	1	B	●	●	↕	↕	⑥	↕
Shift Right Logic	LSR							64	7	2	74	6	3				M 	●	●	R	↕	⑥	↕
	LSRA													44	2	1	A	●	●	R	↕	⑥	↕
	LSRB													54	2	1	B	●	●	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	●	●	↕	↕	R	●
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	●	●	↕	↕	↕	↕
Subract Acmltrs.	SBA													10	2	1	A − B → A	●	●	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	●	●	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	●	●	↕	↕	R	●
	TBA													17	2	1	B → A	●	●	↕	↕	R	●
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	●	●	↕	↕	R	R
	TSTA													4D	2	1	A − 00	●	●	↕	↕	R	R
	TSTB													5D	2	1	B − 00	●	●	↕	↕	R	R

FIGURE 1-3-1 MC6800 Instruction Set

INDEX REGISTER AND STACK POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$(X_H/X_L) - (M/M+1)$	•	•	⑦	↕	⑧	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

JUMP AND BRANCH OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										$C + Z = 0$	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										$C + Z = 1$	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2										See Special Operations	•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1	See special Operations	•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1		•	S	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	⑪	•	•	•	•

CONDITIONS CODE REGISTER OPERATIONS	MNEMONIC	INHER OP	INHER ~	INHER #	BOOLEAN OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	A → CCR	⑫					
CCR → Acmltr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES:

(Bit set if test is true and cleared otherwise)

①	(Bit V)	Test: Result = 10000000?
②	(Bit C)	Test: Result = 00000000?
③	(Bit C)	Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
④	(Bit V)	Test: Operand = 10000000 prior to execution?
⑤	(Bit V)	Test: Operand = 01111111 prior to execution?
⑥	(Bit V)	Test: Set equal to result of $N \oplus C$ after shift has occurred.
⑦	(Bit N)	Test: Sign bit of most significant (MS) byte of result = 1?
⑧	(Bit V)	Test: 2's complement overflow from subtraction of LS bytes?
⑨	(Bit N)	Test: Result less than zero? (Bit 15 = 1)
⑩	(All)	Load Condition Code Register from Stack. (See Special Operations)
⑪	(Bit I)	Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
⑫	(ALL)	Set according to the contents of Accumulator A.

LEGEND:

OP	Operation Code (Hexadecimal);	00	Byte = Zero;
~	Number of MPU Cycles;	H	Half-carry from bit 3;
#	Number of Program Bytes;	I	Interrupt mask
+	Arithmetic Plus;	N	Negative (sign bit)
−	Arithmetic Minus;	Z	Zero (byte)
•	Boolean AND;	V	Overflow, 2's complement
M_{SP}	Contents of memory location pointed to be Stack Pointer;	C	Carry from bit 7
+	Boolean Inclusive OR;	R	Reset Always
⊕	Boolean Exclusive OR;	S	Set Always
$\overline{M}$	Complement of M;	↕	Test and set if true, cleared otherwise
→	Transfer Into;	•	Not Affected
0	Bit = Zero;	CCR	Condition Code Register
		LS	Least Significant
		MS	Most Significant

FIGURE 1-3-1 (continued)

HOW I AVOIDED A SORE FINGER

I have the Microdisk system from North Star Computers. Recently I had a problem with it because of my new keyboard. Both the North Star disk operating system and BASIC accept only upper case letters for commands and for BASIC statements. This was no problem with my first keyboard, as it only produced capital letters. However, my new keyboard produces the full ASCII character set and has no facility for what is variously called "alpha lock" or "caps lock". Extra logic could be added to the keyboard, but I thought the software solution to the problem would be faster for me to implement.

The penalty I have payed is an inability to use lower case letters inside BASIC strings, which does not seem too much to pay for not having to constantly change from upper to lower case and back again. The following fragment of 8080 code (Intel mnemonics) was added to the end of my keyboard input routine. I checks for letters a-z and converts these to A-Z.

```
CPI 61H,  RC,  CPI 7BH,  RNC,  SUI 20H,  RET.
```

Tony Beresford.

COMPUTER TIME TRIALS

When I had received my FORTRAN-80 compiler, I did some simple timing tests to measure the power of a microcomputer as a "number cruncher". I also wished to get a quantitive comparison between a compiler and an interpreter.

My timing technique was simple. I had the computer do a loop up to 1000 times. I first used an empty loop to find the time involved in executing the loop itself and then did several simple calculations inside the loop. Timing was done manually with a stopwatch since I do not have a real-time clock. The time intervals actually involved (from 20 to 200 seconds) meant that errors due to the finite human response time were insignificant.

The comparisons are given below between North Star BASIC and FORTRAN-80. The BASIC has a 12 digit precision (more than adequate for most scientific purposes) while single precision FORTRAN-80 is about 7 digit precision, with double precision being very near to 16 digit precision. Functions not given for North Star were not available.

	North Star	FORTRAN-80
Real addition	4.3	0.5
Real multiplication	8.7	2.4
Real division	28.9	3.8
Double precision addition		2.0
Double precision multiply		5.4
Double precision division		38.7
Sine function	183.3	22.4
Square root	172.3	26.2
Arctangent	216.5	23.8
Double precision sine		239.8

Tony Beresford