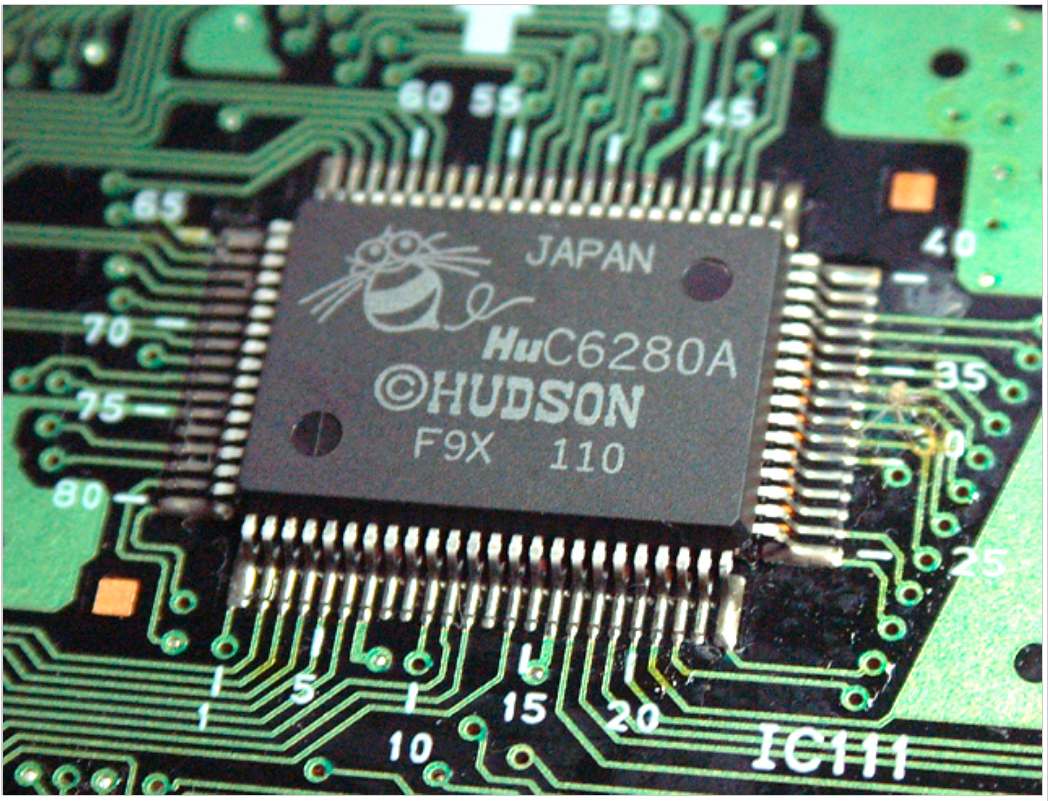
HuC6280

From Archaic Pixels

The Hudson Soft HuC6280 is the 8-bit CPU of NEC's PC Engine console. It contains a customized version of a 65SC02 core, a timer, and sound generation hardware. The HuC6280 is referred to as "Dr. Pepper" or "DRP" by NEC-HE.



The HuC6280

Contents

- 1 Description
- 2 Programmable Sound Generator
- 3 Memory Mapping
- 4 Memory Map
- 5 Instruction Set
- 6 Timer
- 7 Pin assignments
- 8 Patents
  - 8.1 Patent for Memory Access
  - 8.2 Patent for Memory Mapping
  - 8.3 Patent for VRAM access via VDC/CPU
  - 8.4 Patent for Block Transfers Instructions
  - 8.5 Patent for Store Immediate Instructions
  - 8.6 Patent for the Programmable Sound Generator
  - 8.7 Patent for High/Low Speed Modes

Description

The HuC6280 contains a modified | 65C02 (http://en.wikipedia.org/wiki/65C02) core which has several enhancements.

- additional instructions
- instructions to more efficiently write predefined values to the VDC (HuC6270)
- internal peripheral functions such as an interrupt controller
- a Memory Management Unit

- a timer
- an 8-bit parallel I/O port
- a Programmable Sound Generator.

The processor operates at two speeds, 1.7897725 MHz and 7.15909 MHz.

Programmable Sound Generator

Full article: Programmable Sound Generator

The PSG (Programmable Sound Generator) provides 6 sound channels, which can be conveniently paired according to the functionality they provide:

```
    0-1 - Waveform playback
          Frequency modulation (channel 1 muted)
    2-3 - Waveform playback only
    4-5 - Waveform playback
          White noise generation
```

Waveform playback is the most common and allows a 32 byte, 5 bit unsigned linear sample to be played back at selected frequencies. Frequency modulation takes this one step further, allowing the playback frequency to be dynamically adjusted according to a specified pattern. White noise is used to simulate percussion instruments and effects, such as explosions, by means of a pseudo-random square wave.

Alternatively, each channel can be individually switched to Direct D/A mode in which the programmer can send data directly to the sound mixer, allowing more complex sound patterns to be generated, such as speech. Inevitably, this requires more programming effort and CPU time.

Memory Mapping

The HuC6280 has a 64 KB logical address space and a 2 MB physical address space. To access this entire memory space, the HuC6280 uses an MMU (Memory Management Unit) that splits the memory space into a 8 KB page. The logical address space is split as follows:

Each logical 8 KB page (or page) is associated with an 8-bit register (MPR0-7) (Mapped Page Register) that contains the index of the 8 KB page in physical memory to map in this page.

Memory Map

The HuC6280 can address 21 bits (2 MB) of physical memory but uses 16-bit logical addresses (e.g. `LDA $8020`). The 64KB logical address space is split into eight 8 KB pages. The location within this pages is defined by the lower 13 bits of the logical address. The remaining upper three bits are are used to select the MPR. Each 8KB pages (of the 64KB logical addressing space) has a corresponding 8-bit MPR which is used to create a 21 bit (13 lower bits of logical address + 8 bits of corresponding MPR) address sent over the bus.

PhysicalAddress = (LogicalAddress **AND** 0x1FFF) **OR** (MPR[LogicalAddress / 0x2000] * 0x2000)

MPR	Logical Memory Range
0	$0000 - $1FFF
1	$2000 - $3FFF
2	$4000 - $5FFF
3	$6000 - $7FFF
4	$8000 - $9FFF
5	$A000 - $BFFF
6	$C000 - $DFFF
7	$E000 - $FFFF

Only two instructions are used to access these registers:

- Transfer Accumulator to MPRi (TAMi)
- Transfer MPRi to Accumulator (TMAi)

Instruction Set

Quick Instruction Set List	Full Instruction Set							
adc	Add With Carry (ADC)	and	And Accumulator with Memory (AND)	asl	Accumulator Shift Left (ASL)	bbri	Branch on Bit Reset (BBRi)	bbsi	Branch on Bit Set (BBSi)
bcc	Branch on Carry Clear (BCC)	bcs	Branch on Carry Set (BCS)	beq	Branch on Equal (BEQ)	bit	Test Memory Bits against Accumulator (BIT)	bmi	Branch on Minus (BMI)
bne	Branch on Not Equal (BNE)	bpl	Branch on Plus (BPL)	bra	Branch Always (BRA)	brk	Break (BRK)	bsr	Branch to Subroutine (BSR)
bvc	Branch if Overflow Clear (BVC)	bvs	Branch if Overflow Set (BVS)	cla	Clear Accumulator (CLA)	clc	Clear Carry Flag (CLC)	cld	Clear Decimal Flag (CLD)
cli	Clear Interrupt Disable Flag (CLI)	clv	Clear Overflow Flag (CLV)	clx	Clear X Register (CLX)	cly	Clear Y Register (CLY)	cmp	Compare Accumulator with Memory (CMP)
cpx	Compare X Register with Memory (CPX)	cpy	Compare Y Register with Memory (CPY)	csh	Change Speed High (CSH)	csl	Change Speed Low (CSL)	dec	Decrement (DEC)
dex	Decrement X (DEX)	dey	Decrement Y (DEY)	eor	Exclusive OR Accumulator with Memory (EOR)	inc	Increment (INC)	inx	Increment X (INX)
iny	Increment Y (INY)	jmp	Jump (JMP)	jsr	Jump to Subroutine (JSR)	lda	Load Accumulator from Memory (LDA)	ldx	Load X Register from Memory (LDX)
ldy	Load Y Register from Memory (LDY)	lsr	Logical Shift Memory or Accumulator Right (LSR)	nop	No Operation (NOP)	ora	Or Accumulator with Memory (ORA)	pha	Push Accumulator (PHA)
php	Push Processor Status Register (PHP)	phx	Push X Register (PHX)	phy	Push Y Register (PHY)	pla	Pull Accumulator (PLA)	plp	Pull Processor Status Register (PLP)
plx	Pull X Register (PLX)	ply	Pull Y Register (PLY)	rmbi	Reset Memory Bit i (RMBi)	rol	Rotate Memory or Accumulator Left (ROL)	ror	Rotate Memory or Accumulator Right (ROR)
rti	Return from Interrupt (RTI)	rts	Return from Subroutine (RTS)	sax	Swap Accumulator and X	say	Swap Accumulator and Y	sbc	Subtract with Borrow from

				Register (SAX)		Register (SAY)		Accumulator (SBC)
sec	Set Carry Flag (SEC)	sed	Set Decimal Mode Flag (SED)	sei	Set Interrupt Disable Flag (SEI)	set	Set T Flag (SET)	smbi	Set Memory Bit i (SMBi)
st0	Store HuC6270 No. 0 (ST0)	st1	Store HuC6270 No. 1 (ST1)	st2	Store HuC6270 No. 2 (ST2)	sta	Store Accumulator to Memory (STA)	stx	Store X Register to Memory (STX)
sty	Store Y Register to Memory (STY)	stz	Store Zero to Memory (STZ)	sxy	Swap X and Y Registers (SXY)	tai	Transfer Alternate Increment (TAI)	tami	Transfer Accumulator to MPRi (TAMi)
tax	Transfer Accumulator to X Register (TAX)	tay	Transfer Accumulator to Y Register (TAY)	tdd	Transfer Decrement Decrement (TDD)	tia	Transfer Increment Alternate (TIA)	tii	Transfer Increment Increment (TII)
tin	Transfer Increment None (TIN)	tmai	Transfer MPRi to Accumulator (TMAi)	trb	Test and Reset Memory Bits Against Accumulator (TRB)	tsb	Test and Set Memory Bits Against Accumulator (TSB)	tst	Test and Reset Memory Bits (TST)
tsx	Transfer Stack Pointer to X Register (TSX)	txa	Transfer X Register to Accumulator (TXA)	txs	Transfer X Register to Stack Pointer (TXS)	tya	Transfer Y Register to Accumulator (TYA)		

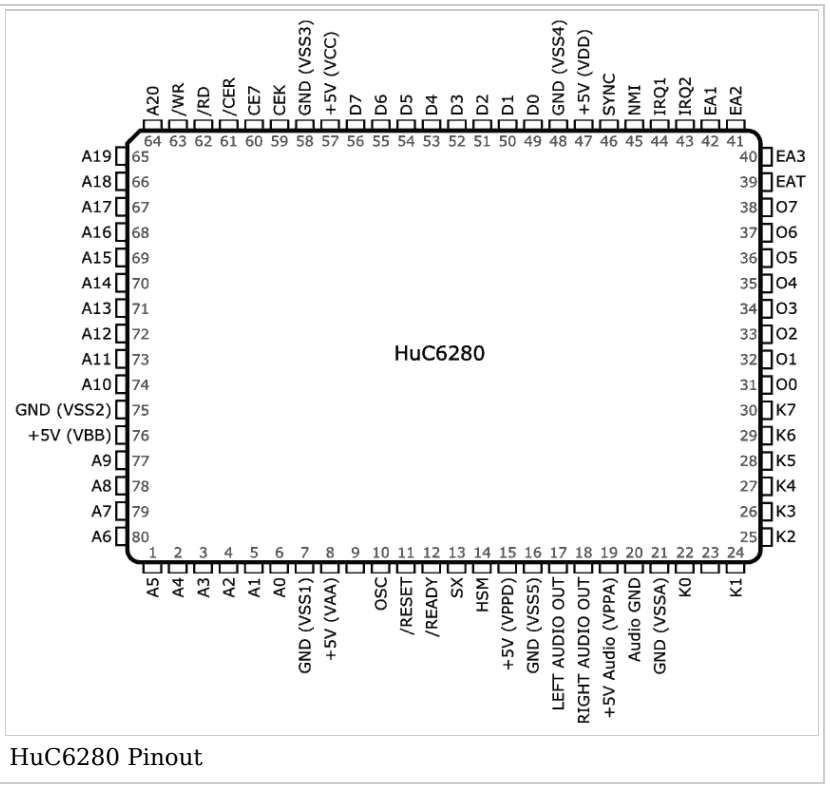
Timer

The HuC6280's timer operates off of the 7.15909 MHz clock. This clock is first sent into a divide-by-1024 counter, and the output of the clock divider is used to decrement the timer counter register, if the timer is enabled. When the timer counter register decrements when its value is 0, the timer counter will reload with the value contained in the timer latch, and the timer's IRQ line is activated. The timer's IRQ vector is located at logical address $FFFA. The timer IRQ can be acknowledged(IE the timer's IRQ line made inactive again to prevent future interrupts) by reading from the IRQ mask register, or writing to the read-only IRQ status register.

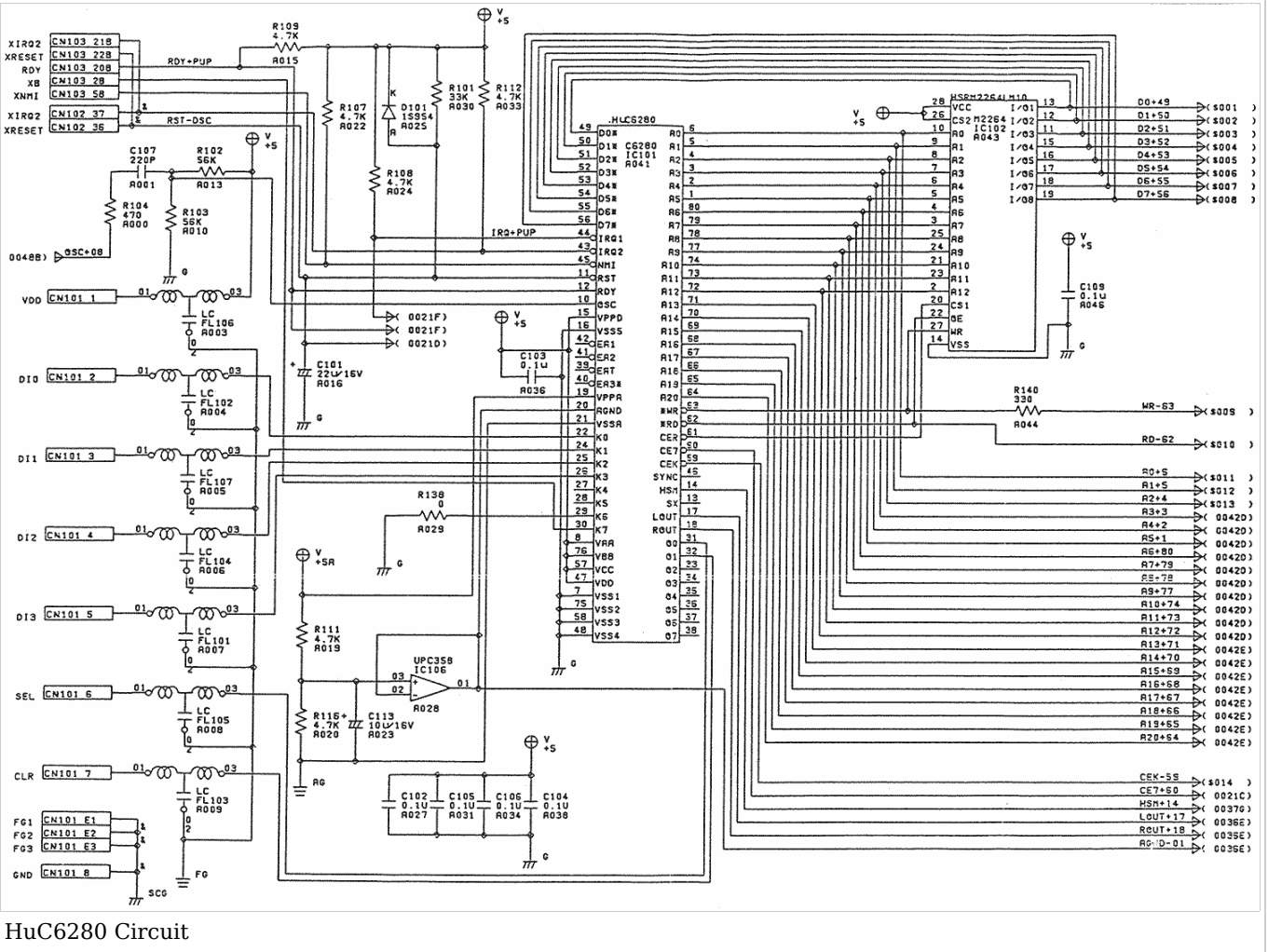
Address(in I/O page)	R/W	Bits	Description
$0C00	R	Bit 0 - 6	Current timer counter value.
		Bit 7	(Undefined, I/O data buffer D7)
	W	Bit 0 - 6	Timer latch value.
		Bit 7	Unused
$0C01	W	Bit 0	Timer enabled if set to 1. Writing 0, then 1, will force a reload of the timer counter from the timer latch and reset the divide-by-1024 counter.
		Bit 1 - 7	Unused

Pin assignments

Pin	Signal	Direction	Description



HuC6280 Pinout



1	A5	out	Address bus, bit 5
2	A4	out	Address bus, bit 4
3	A3	out	Address bus, bit 3
4	A2	out	Address bus, bit 2

5	A1	out	Address bus, bit 1
6	A0	out	Address bus, bit 0
7	GND	s	Ground (VSS1)
8	+5V	s	Power Supply (VAA)
9	XOUT	out	Follows XIN polarity (different pulse shape) (Not Connected on PC Engine)
10	XIN	in	21.477270 MHz clock input (OSC1)
11	$\overline{\text{RESET}}$	in	Reset signal input
12	RDY	in	Induce wait state while pulled low
13	SX	out	Complementary CPU clock output (1 = 7.15909, 0 = 1.7897725 MHz)
14	HSM	out	High Speed Mode (1 = 7.15909, 0 = 1.7897725 MHz)
15	+5V	s	Power Supply (VPDD)
16	GND	s	Ground (VSS5)
17	LOUT	out	Audio output, Left channel
18	ROUT	out	Audio output, Right channel
19	+5V	s	Audio Power (VPPA)
20	+2.5V	s	Power Supply (VEE)
21	AGND	s	Ground
22	K0	in	Input port, bit 0 (@FF:1000)
23		out	Unknown/Undocumented (combination of SX, BSY, A0?) (Not Connected on PC Engine)
24	K1	in	Input port, bit 1 (@FF:1000)
25	K2	in	Input port, bit 2 (@FF:1000)
26	K3	in	Input port, bit 3 (@FF:1000)
27	K4	in	Input port, bit 4 (@FF:1000)
28	K5	in	Input port, bit 5 (@FF:1000)
29	K6	in	Input port, bit 6 (@FF:1000)
30	K7	in	Input port, bit 7 (@FF:1000)
31	O0	out	Output port, bit 0 (@FF:1000)
32	O1	out	Output port, bit 1 (@FF:1000)
33	O2	out	Output port, bit 2 (@FF:1000)
34	O3	out	Output port, bit 3 (@FF:1000)
35	O4	out	Output port, bit 4 (@FF:1000)
36	O5	out	Output port, bit 5 (@FF:1000)
37	O6	out	Output port, bit 6 (@FF:1000)
38	O7	out	Output port, bit 7 (@FF:1000)
39	$\overline{\text{EAT}}$	out	Not Connected on PC Engine
40	$\overline{\text{EA3}}$	out	Not Connected on PC Engine
41	$\overline{\text{EA2}}$	out	Not Connected on PC Engine
42	$\overline{\text{EA1}}$	out	Not Connected on PC Engine
43	$\overline{\text{IRQ2}}$	in	$\overline{\text{IRQ2}}$ interrupt input
44	$\overline{\text{IRQ1}}$	in	$\overline{\text{IRQ1}}$ interrupt input
45	$\overline{\text{NMI}}$	in	$\overline{\text{NMI}}$ interrupt input
46	SYNC	out	Memory read type; (1 = Opcode fetch, 0 = Not opcode fetch)
47	+5V	s	Power Supply (VDD)
48	GND	s	Ground (VSS4)
49	D0	in / out	Data bus, bit 0
50	D1	in / out	Data bus, bit 1
51	D2	in / out	Data bus, bit 2
52	D3	in / out	Data bus, bit 3
53	D4	in / out	Data bus, bit 4
54	D5	in / out	Data bus, bit 5
55	D6	in / out	Data bus, bit 6
56	D7	in / out	Data bus, bit 7
57	+5V	s	Power supply (VCC)

58	GND	s	Ground
59	$\overline{\text{CEK}}$	out	HuC6260 $\overline{\text{CS}}$ (@ FF:0400-0700)
60	$\overline{\text{CE7}}$	out	HuC6270 $\overline{\text{CS}}$ (@ FF:0000-03FF)
61	$\overline{\text{CER}}$	out	Work RAM $\overline{\text{CS}}$ (@ F8:0000-1F00)
62	$\overline{\text{RD}}$	out	Memory read strobe
63	$\overline{\text{WR}}$	out	Memory write strobe
64	A20	out	Address bus, bit 20
65	A19	out	Address bus, bit 19
66	A18	out	Address bus, bit 18
67	A17	out	Address bus, bit 17
68	A16	out	Address bus, bit 16
69	A15	out	Address bus, bit 15
70	A14	out	Address bus, bit 14
71	A13	out	Address bus, bit 13
72	A12	out	Address bus, bit 12
73	A11	out	Address bus, bit 11
74	A10	out	Address bus, bit 10
75	GND	s	Ground (VSS2)
76	+5V	s	Power Supply (VBB)
77	A9	out	Address bus, bit 9
78	A8	out	Address bus, bit 8
79	A7	out	Address bus, bit 7
80	A6	out	Address bus, bit 6

Patents

- Memory Access (https://www.google.com/patents/US4970642)
- Memory Mapping (https://www.google.com/patents/US5566313)
- VRAM access by VDC/CPU (https://www.google.com/patents/US5030946)
- Block Transfers Instructions (https://www.google.com/patents/US5034886)
- Store Immediate Instructions (https://www.google.com/patents/US5226140)
- High/Low Speed Modes (https://www.google.com/patents/US5483659)
- Programmable Sound Generator (https://www.google.com/patents/US4924744)

Patent for Memory Access

Note: terminals with a *High* value are a binary *1* and *Low* is a binary *0*.

Full Patent

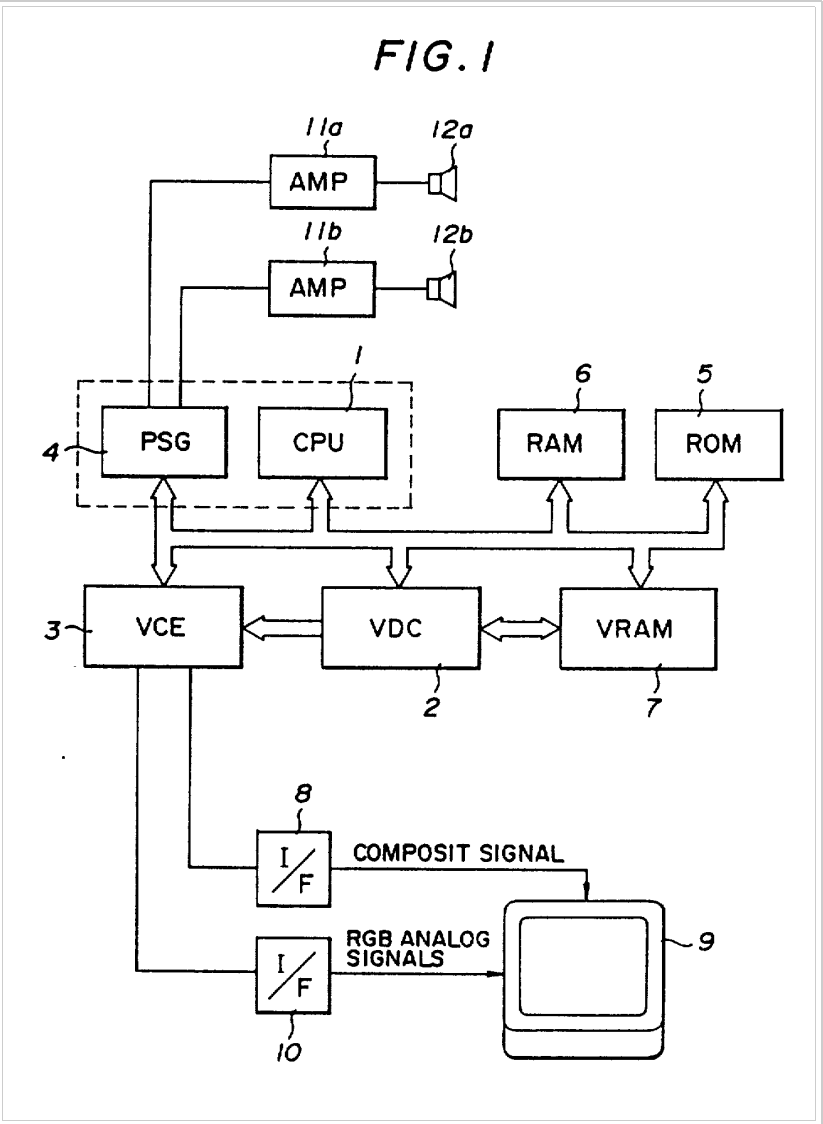
Original Patent (https://www.google.com/patents/US4970642)

United States Patent Number: **4970642**

AN APPARATUS FOR ACCESSING A MEMORY

In [Figure 1], there is shown an apparatus for displaying a color image to which an apparatus for controlling the access of a memory according to the invention is applied In the apparatus for controlling the access of a memory, a *CPU* (1) performs a predetermined control in accordance with a program stored in *ROM* (5) so that data, arithmetical results etc. are stored into a *RAM* (6) temporarily. A *Video Display Controller* (2) is provided therein to supply a *Video Color Encoder* (3) with video data of a story, for instance, for a so-called television game read from a V*RAM* (7) in accordance with a control of the *CPU* (1) which deciphers a program for the television game stored in the *ROM* (5). The

Video Color Encoder (3) to which the video data are supplied produces RGB analog signals obtained in accordance with color data stored therein, or produces video color signal including a luminance signal and color difference signals obtained in accordance with the color data. Further, a *Programmable Sound Generator* (4) is provided therein to produce analog sound signals as left and right stereo sounds in accordance with a content of the *ROM* (5) which is supplied through the *CPU* (1) thereto. The video color signal produced in the *Video Color Encoder* (3) is supplied through an *interface* (8) to a receiving circuit of a *video display* (9) as a composite signal, and the RGB analog signal is supplied through an *interface* (10) directly to a *video display* (9) which functions as an exclusive use monitor means. On the other hand, the left and right analog sound signals are supplied through amplifiers 11a and 11b to speakers 12a and 12b to produce sounds.



[Figure 1] is a block diagram showing an apparatus for displaying a color image in which an apparatus for controlling the access of a memory is included

[Figure 2] shows the *CPU* (1) and the *Programmable Sound Generator* (4) as encircled by a dotted line in [Figure 1]. The *CPU* (1) in which an apparatus for controlling a transfer of data in the embodiment is included and comprises an *instruction register* (20), an *instruction decoder* (21), a *bus interface register* (22), an *Arithmetic Logic Unit* (ALU) (23), a *set of registers* (24), a *Mapping Register* (25), a *chip enable decoder* (26), a timing and *control unit* (27), an *input and output port* (28), a *Timer* (29), an *interrupt request register* (30), an *interrupt disable register* (31), and so on. These units will be explained as follows.

1. *instruction register* (20)

   The register (20) is loaded with an instruction code at an instruction fetch cycle.

2. *instruction decoder* (21)

   The decoder (21) performs a sequential operation determined in accordance with an output of the *instruction register* (20), an interrupt input from a peripheral circuit or a reset input, and further performs a control of a branch command changing a flow of a program in accordance with informations of a status register described later.

3. *bus interface register* (22)

The register (22) controls a transfer of data among a *B-bus* (32), a *U-bus* (33) and an external bus D0 to D7. The *ALU* (23) and the *set of registers* (24) are connected by the *B-bus* (22) and the *U-bus* (33), and is connected to internal peripheral circuits. Further, a *L-bus* (34) for transferring lower 8 bits of a logical address and a *H-bus* (35) for transferring upper 8 bits of the logical address are provided. A *logical address low register* (48) is connected to the *L-bus* (34), and a *logical address high register* (49) is connected to the *H-bus* (35).

4. *ALU* (23)

The *ALU* (23) is provided with an *A register* (36) and a *B register* (37), and performs all of arithmetic and logic operation. The A and B registers (36 and 37) are loaded with one or two data so that an arithmetic operation is performed in accordance with a control signal of the *instruction decoder* (21) to supply one of the b, L and H-buses (32, 34 and 35) with a result of the arithmetic operation.

5. *set of registers* (24)

The *set of registers* (24) comprises following 10 registers each being of 8 bits.

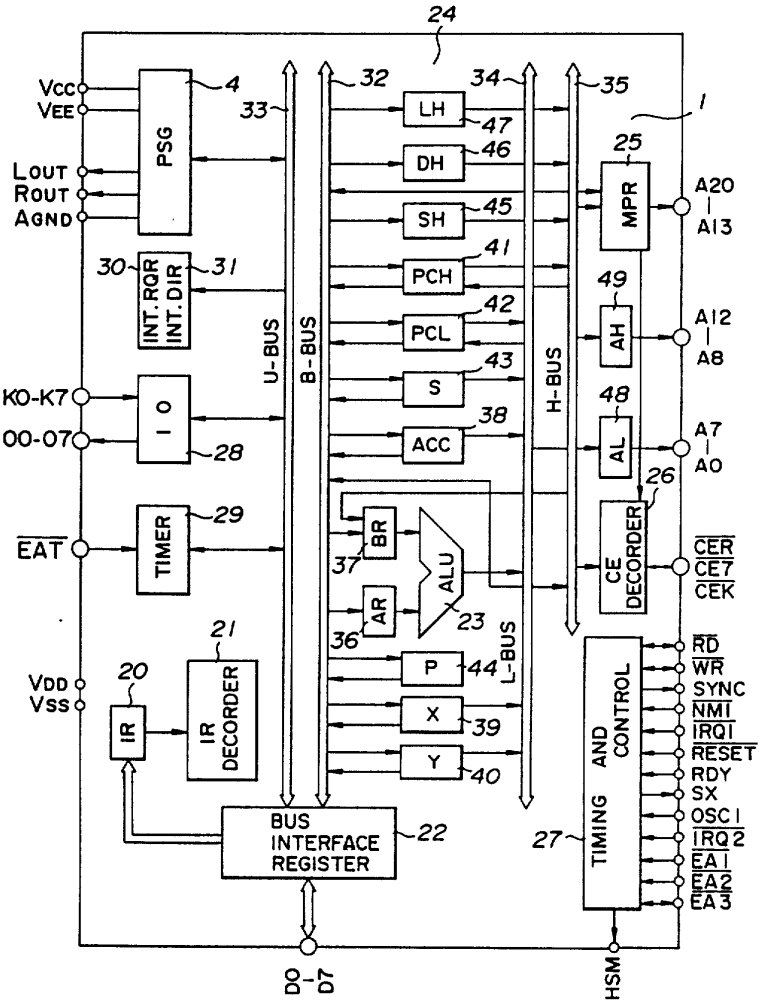
a. *Accumulator* (38)

The *Accumulator* (38) is a wide use register which plays the most important role in an arithmetic and logic operation to be conducted when a memory arithmetic flag T of a status register described later is *0*. Data thereof is supplied to an input of the *ALU* (23), and a result of the arithmetic is stored therein. The *Accumulator* (38) is also used for a transfer of data between memories and between a memory and a peripheral circuit, and for a count of a data block length when a block transfer of data is performed. A lower data of the length are stored therein after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6).

b. X and Y registers (39 and 40)

The registers (39 and 40) are wide use registers which are mainly used for an index addressing. The *X register* (39) is used for a designation of an address on page *0* of a memory which is a destination of an arithmetic operation, and for a storage of lower data of a source address after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed. On the other hand, the *Y register* (40) stores lower data of a destination address after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed.

FIG.2



[Figure 2] is a block diagram showing an apparatus for controlling the access of a memory in an embodiment according to the invention

c. program counters (41 and 42)

An up counter of 16 bits is composed of the *Program Counter* (41) of upper 8 bits and the *Program Counter* (42) of lower 8 bits. The up counter is automatically incremented in accordance with the conduct of a command to designate an address of a command or operand to be next conducted. Contents of the counters (41 and 42) are evacuated into a stack region of the *RAM* (6) in a case where a command of subroutine is conducted, and an interrupt is produced, or after an interruption command of a software is conducted.

d. *Stack Pointer* (43)

The *Stack Pointer* (43) designates lower 8 bits of the highest address on a stack region of the *RAM* (6), and is decremented after the pushing of data into the stack region and incremented before the pulling of the data from the stack region. For instance, 256 bytes of addresses *0x2100* to *0x21FF* are allocated to the stack region in a logical address.

e. *source high register* (45), *destination high register* (46), and *length high register* (47).

These registers function in case of a command of a block transfer. The *source high register* (45) provides an upper byte of a source address to designate the source address together with a content of the *X register* (39). The *destination high register* (46) provides an upper byte of a destination address to designate the destination address together with a content of the *Y register* (40). The *length high register* (47) provides upper 8 bits for a down counter together with a content of the *Accumulator* (38) so that a length of a block transfer is counted by a byte unit.

6. *Mapping Register* (25)

The *Mapping Register* (25) is composed of 8 registers each being of 8 bits to convert a logical address of 16 bits to a physical address of 21 bits, and is selected by upper 3 bits of the *H-bus* (35).

7. *chip enable decoder* (26)

The *chip enable decoder* (26) provides chip enable outputs for following peripheral circuits by decoding upper 11 bits of a physical address.

a. a chip enable for the *RAM* (6) . . . $\overline{CER}$

b. a chip enable for the *Video Display Controller* (2) . . . $\overline{CE7}$

c. a chip enable for the *Video Color Encoder* (3) . . . $\overline{CEK}$

d. a chip enable for the *Programmable Sound Generator* (4) . . . $\overline{CEP}$

e. a chip enable for the *Timer* (29) . . . $\overline{CET}$

f. a chip enable for the input and

output port . . . $\overline{CEIO}$

g. a chip enable for the *interrupt request register* (30) and the *interrupt disable register* (31) . . . $\overline{CECG}$

8. timing and *control unit* (27)

The unit (27) is connected to following terminals.

a. $\overline{RD}$ terminal

A read timing signal is supplied through the $\overline{RD}$ terminal at a reading cycle.

b. $\overline{WR}$ terminal

A write timing signal is supplied through the $\overline{WR}$ terminal at a writing cycle.

c. SYNC terminal

A synchronous signal of *High* is supplied through the SYNC terminal at an instruction fetch cycle, that of *Low* is supplied therethrough at a system reset timing.

d. $\overline{NMI}$ terminal

A non-maskable interrupt is produced when $\overline{NMI}$ input signal is supplied through the $\overline{NMI}$ terminal. A sub-routine call is conducted by reading lower address from the logical address *0xFFFC* and upper address from the logical address *0xFFFD* when a command which is conducted in a program is completed.

e. $\overline{IRQ1}$ and $\overline{IRQ2}$ terminals

A sub-routine call is conducted by reading lower address from the logical address *0xFFF8* and upper address from the logical address *0xFFF9* when $\overline{IRQ1}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

A sub-routine call is conducted by reading lower address from the logical address *0xFFF6* and upper address from the logical address *0xFFF7* when $\overline{IRQ2}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

f. $\overline{RESET}$ terminal

A program is started by reading lower address from the physical address *0x001FFE* and upper address from the physical address *0x001FFF* when a $\overline{RESET}$ input becomes *Low*.

g. RDY terminal

The *CPU* (1) is started to operate when a RDY input is changed from *Low* to *High*.

h. SX terminal

A complementary signal of a system clock signal is supplied through the SX terminal.

i. OSC1 terminal

An external clock signal is input through the OSC1 terminal.

j. $\overline{EA1}$, $\overline{EA2}$ and $\overline{EA3}$ terminals

These are input terminals for a test of the *CPU* (1).

k. HSM terminal

A speed signal of *High* is supplied through the HSM terminal in case of a high speed mode of 21.47727 MHz/3, and that of *Low* is supplied therethrough in case of a low speed mode of 21.47727 MHz/12.

9. *input and output port* (28)

The *input and output port* (28) is connected to following terminals.

a. K0 to K7 terminals

The terminals are input ports from which data are written in accordance with the conduct of a reading cycle in regard to the physical addresses *0x1FF000* to *0x1FF3FF*.

b. $\overline{00}$ to $\overline{07}$ terminals

The terminals are output ports with latches to which data are supplied in accordance with the conduct of a writing cycle in regard to the physical addresses *0x1FF000* to *0x1FF3FF*.

10. *Timer* (29)

The *Timer* (29) is connected to a test input terminal $\overline{EAT}$ for the *CPU* (1) and provides a timer signal through the U-bus thereto.

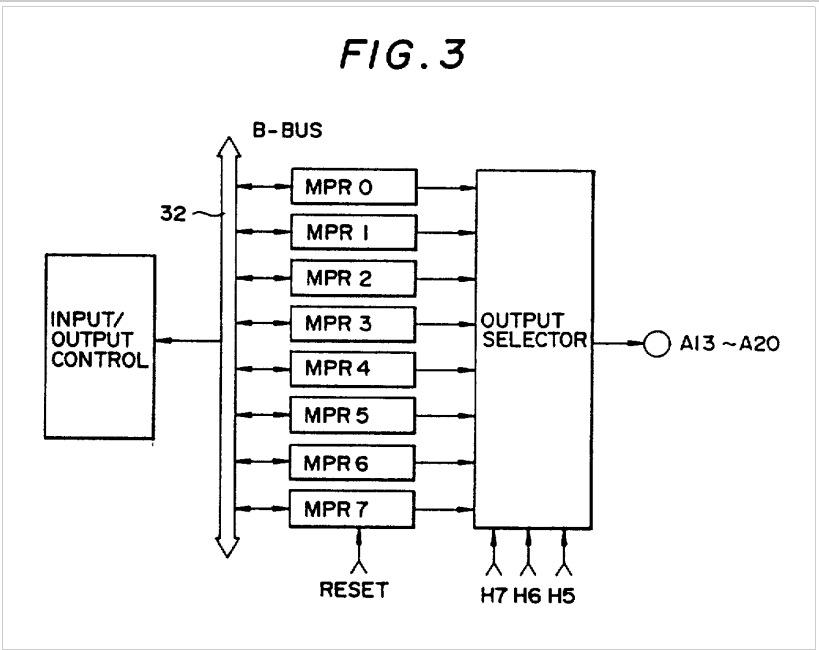
11. *interrupt request register* (30)

The register (30) is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *1* to show the $\overline{IRQ1}$ and $\overline{IRQ2}$ terminals *Low* and the remaining 1 bit is *1* to show a timer interrupt caused. The register (30) is only used for *read*.

12. *interrupt disable register* (31)

The register (31) is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *0* to make an interrupt request of the $\overline{IRQ1}$ and $\overline{IRQ2}$ terminals disable, and the remaining one is *0* to make an interrupt request disable in accordance with the timer interrupt signal.

In [Figure 3], there is shown the aforementioned *Mapping Register* (25) comprising 8 registers MPR0 to MPR7 each being of 8 bits, and connected through the *B-bus* (32) to an *input output controller* (50) and through an *output selector* (51) to the output terminals A13 to A20. The *output selector* (51) selects one register from the 8 registers MPR0 to MPR7 of the *Mapping Register* (25) in accordance with upper 3 bits H5 to H7 of upper data of a logical address on the *H-bus* (35).



[Figure 3] is a block diagram showing a mapping register circuit in an apparatus for controlling the access of a memory in the embodiment

[Figure 4] shows a relation between the upper 3 bits H5 to H7 and one register selected from the 8 registers MPR0 to MPR7. If it is assumed that the upper 3 bits H5 to H7 are *010*, the register MPR2 is selected from the 8 registers MPR0 to MPR7. Data are read from the *Mapping Register* (25) by a command TMAi where *i* is an integer selected from *0* to *7*. For instance, data are read from the register MPR2 to be transferred through the *B-bus* (32) to the *Accumulator* (38) in accordance with a command TMA2. On the other hand, data are written into the *Mapping Register* (25) by a command TAMi where *i* is an integer selected from *0* to *7*. For instance, data are transferred to be written into the register MPR0 from the *Accumulator* (38) by a command TAM0. The commands TMAi and TAMi are composed of two byte respectively, and lower byte thereof includes a bit of *1* corresponding in a bit number to a register number which is selected from the 8 registers MPR0 to MPR7 and remaining 7 bits of *0*. When one of the 8 registers MPR0 to MPR7 is selected in accordance with upper 3 bits of the *H-bus* (35), a content of the selected register is supplied through the output terminals A13 to A20 to a following stage so that a physical address of 21 bit is obtained together with a content of the *logical address low register* (48) to be supplied through the output terminals A0 to A7 thereto, and lower 5 bits of the *logical address high register* (49) to be supplied through the output terminals A8 to A12 thereto. When the most significant bit A20 of a physical address A0 to A20 is *1*, a command by which data designated in accordance with the physical address A0 to A20 are immediate-transferred to the *Video Display Controller* (2) is conducted. The command includes ST0, ST1 and ST2 by which codes are produced on A0 and A1 of the address bus at a write cycle as shown in [Figure 6]. The command is set in the *instruction register* (20).

In operation, lower data of a logical address are set in the logic address low register (48), and lower 5 bits of upper data of the logical address are set in the *logical address high register* (49). Here, it is assumed that address data are set in the 8 registers MPR0 to MPR7 of the *Mapping Register* (25) respectively. When upper 3 bits of upper data of the logical address, that is to say, upper 3 bits H5 to H7 of the *H-bus* (35) are *001*, the register MPR1 is selected so that a content *0xF8* of the register MPR1 is

FIG.4

H7	H6	H5		H7	H6	H5	
0	0	0	MPR0	1	0	0	MPR4
0	0	1	MPR1	1	0	1	MPR5
0	1	0	MPR2	1	1	0	MPR6
0	1	1	MPR3	1	1	1	MPR7

[Figure 4] is an explanatory diagram showing a mapping register selection code when a physical address is produced in an apparatus for controlling the access of a memory in the embodiment

supplied through the output terminals A13 to A20 to the following stage. These output signals are combined with output signals of the output terminals A8 to A12 and A0 to A7 to produce a physical address A0 to A20. Then, the *chip enable decoder* (26) decodes upper 11 bits A10 to A20 of the physical address A0 to A20 to produce a chip enable signal $\overline{CER}$ of *0* by which a data memory is enabled.

FIG.5

MAPPING REGISTER	LOWER BYTE (BINARY) MSB LSB
MPR0	00000001
MPR1	00000010
MPR2	00000100
MPR3	00001000
MPR4	00010000
MPR5	00100000
MPR6	01000000
MPR7	10000000

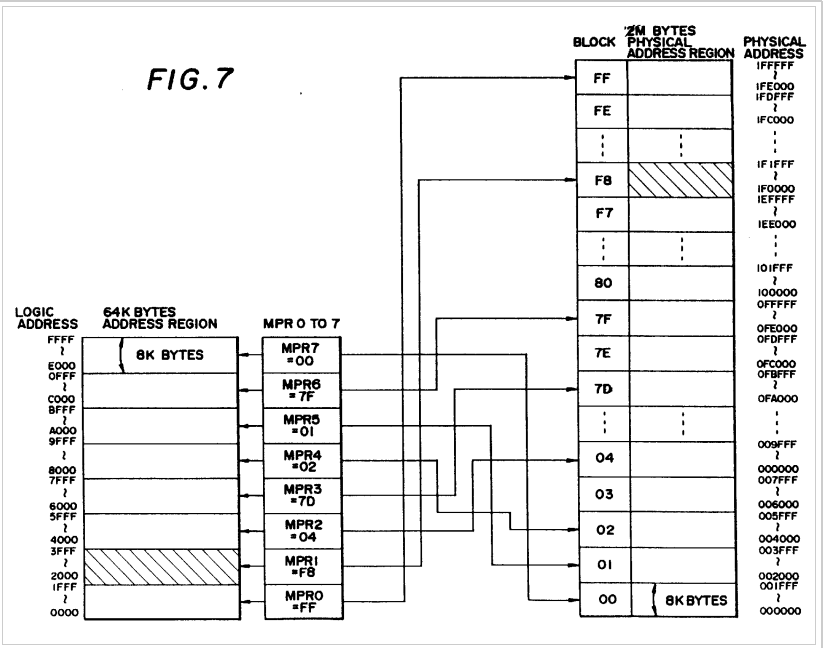
[Figure 5] is an explanatory diagram showing a mapping register selection code when data are read from a mapping register and written thereinto in an apparatus for controlling the access of a memory in the embodiment

FIG.6

IR \ ADDRESS	A1	A0
ST0	0	0
ST1	1	0
ST2	1	1

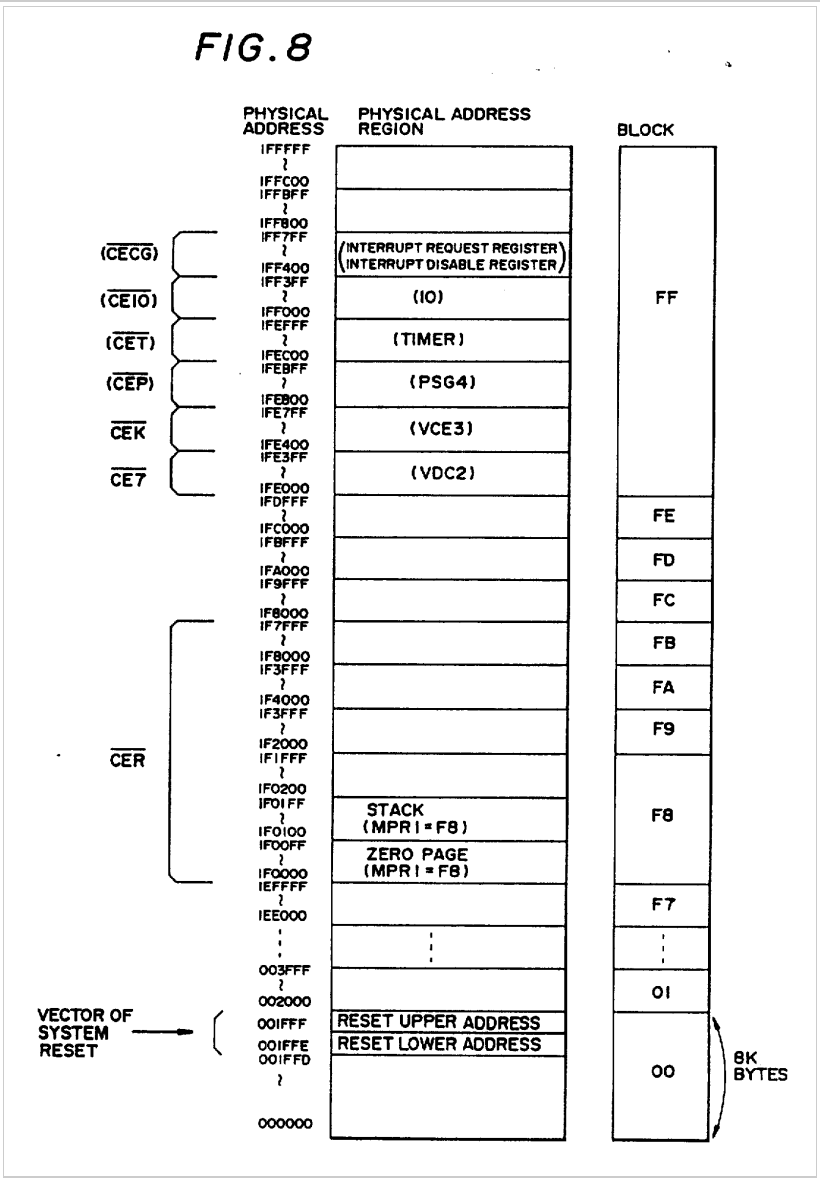
[Figure 6] is an explanatory diagram showing an immediate transfer code for a video display controller in the apparatus for displaying a color image

[Figure 7] shows a relation between a logical address of the *CPU* (1) and a physical address of the *RAM* (6) wherein a block *0xF8* of a physical address region having 2M bytes is allocated in accordance with a content *0xF8* of the selected register MPR1 in a case where a logic address of 16 bits on the L and H-buses (34 and 35) corresponds to one address selected from addresses *0x2000* to *0x3FFF* of a logical address region having 64K bytes as indicated by hatching lines therein. Thus, a memory of 2M bytes in which an address signal of 21 bits is required for the access thereof can be accessed by an address signal of 16 bits. At this time, data at a corresponding address of a memory are immediate-transferred to a *Video Display Controller* (2) in a case where lower 2 bits A0 and A1 of the physical address are of a content as shown in [Figure 6].



[Figure 7] is an explanatory diagram showing a relation between logical and physical address regions in an apparatus for controlling the access of a memory in the embodiment

[Figure 8] shows a physical address region of the *RAM* (6) to which a chip enable signal $\overline{CER}$ is allocated. The chip enable signal $\overline{CER}$ is produced in accordance with a content of upper 11 bits A10 to A20 of a physical address A0 to A20 which is decoded in the *chip enable decoder* (26). For instance, the chip enable signal $\overline{CER}$ is allocated to a region of physical addresses *0x1F0000* to *0x1F7FFF* in which zero page and stack regions are existed in a case where a content of the selected register MPR1 is *0xF8*. The zero page and stack regions are of a region to which contents of the *Accumulator* (38), the *X register* (39), and the *Y register* (40) are evacuated temporarily.



[Figure 8] is an explanatory diagram showing an allocation of a chip enable signal to a physical address region in an apparatus for controlling the access of a memory in the embodiment.

Patent for Memory Mapping

Full Patent

Original Patent (https://www.google.com/patents/US5566313)

United States Patent Number: **5566313**

APPARATUS FOR CONTROLLING THE TRANSFER OF DATA

In [Figure 1] , there is shown an apparatus for displaying a color image to which an apparatus for controlling the transfer of data according to the invention is applied. In the apparatus for controlling the transfer of data, a *CPU* (1) performs a predetermined control in accordance with a program stored in *ROM* (5) so that data, arithmetical results etc. are stored into a *RAM* (6) temporarily. A *Video Display Controller* (2) is provided therein to supply a *Video Color Encoder* (3) with video data of a story, for instance, for a so-called television game read from a video RAM (VRAM) 7 in accordance with a control of the *CPU* (1) which deciphers a program for the television game stored in the *ROM* (5). The *Video Color Encoder* (3) to which the video data are supplied produces RGB analog signals obtained in accordance with color data stored therein, or produces video color signal including a luminance signal and color difference signals obtained in accordance with the color data. Further, a programable sound generator 4 is provided therein to produce analog sound signals as left and right stereo sounds in accordance with a content of the *ROM* (5) which is supplied through the *CPU* (1) thereto. The video color signal produced in the *Video Color Encoder* (3) is supplied through

File:US5566313-fig.png
[Figure 1] is a block diagram showing an apparatus for displaying a color image in which an apparatus for controlling the transfer of data is included.

an *interface* (8) to a receiving circuit of a *video display* (9) as a composite signal, and the RGB analog signal is supplied through an *interface* (10) directly to a *video display* (9) which functions as an exclusive use monitor means. On the other hand, the left and right analog sound signals are supplied through amplifiers 11a and 11b to speakers 12a and 12b to produce sounds.

[Figure 2] shows the *CPU* (1) and the programable sound generator 4 as encircled by a dotted line in [Figure 1]. The *CPU* (1) in which an apparatus for controlling a transfer of data in the embodiment is included and comprises an *instruction register* (20), an *instruction decoder* (21), a *bus interface register* (22), an *Arithmetic Logic Unit* (ALU) (23), a *set of registers* (24), a *Mapping Register* (25), a *chip enable decoder* (26), a *timing and control unit* (27), an *input and output port* (28), a *Timer* (29), an *interrupt request register* (30), an *interrupt disable register* (31), and so on. These units will be explained as follows.

File:US5566313-fig.png
[Figure 2] is a block diagram showing an apparatus for controlling the transfer of data in an embodiment according to the invention.

(1) *instruction register* (20)

The *register* (20) is loaded with an instruction code at an instruction fetch cycle.

(2) *instruction decoder* (21)

The decoder 21 performs a sequential operation determined in accordance with an output of the *instruction register* (20), an interrupt input from a peripheral circuit or a reset input, and further performs a control of a branch command changing a flow of a program in accordance with informations of a status register described later.

(3) *bus interface register* (22)

The *register* (22) controls a transfer of data among a *B-bus* (32), a *U-bus* (33) and an external bus D0 to D7. The *ALU* (23) and the *set of registers* (24) are connected by the *B-bus* (32) and the *U-bus* (33), and is connected to internal peripheral circuits. Further, an *L-bus* (34) for transferring lower 8 bits of a logical address and a *H-bus* (35) for transferring upper 8 bits of the logical address are provided. A *logical address low register* (48) is connected to the *L-bus* (34), and a *logical address high register* (49) is connected to the *H-bus* (35).

(4) *ALU* (23)

The *ALU* (23) is provided with an *A register* (36) and a *B register* (37), and performs all of arithmetic and logic operation. The A and B registers 36 and 37 are loaded with one or two data so that an arithmetic operation is performed in accordance with a control signal of the *instruction decoder* (21) to supply one of the B, L and H-buses 32, 34 and 35 with a result of the arithmetic operation.

(5) *set of registers* (24)

The *set of registers* (24) comprises the following 10 registers each being of 8 bits.

(a) *Accumulator* (38)

The *Accumulator* (38) is a wide use register which plays the most important role in an arithmetic and logic operation to be conducted when a memory arithmetic flag T of a status register described later is *0*. Data thereof is supplied to an input of the *ALU* (23), and a result of the arithmetic is stored therein. The *Accumulator* (38) is also used for a transfer of data between memories and between a memory and a peripheral circuit, and for a count of a data block length when a block transfer of data is performed. A lower data of the length are stored therein after data stored therein are evacuated into a stack region of the *RAM* (6).

(b) X and Y registers 39 and 40

The registers 39 and 40 are wide use registers which are mainly used for an index addressing. The *X register* (39) is used for a designation of an address on page *0* of a memory which is a destination of an arithmetic operation, and for a storage of lower data of a source address after data stored therein are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed. On the other hand, the *Y register* (40) stores lower data of a destination address after data stored therein are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed.

(c) program counters 41 and 42

An up counter of 16 bits is composed of the *Program Counter* (41) of upper 8 bits and the *Program Counter* (42) of lower 8 bits. The up counter is automatically incremented in accordance with the conduct of a command to designate an address of a command or operand to be next conducted. Contents of the counters 41 and 42 are evacuated into a stack region of the *RAM* (6) in a case where a command of sub-routine is conducted, and an interrupt is produced, or after an interruption command of a software is conducted.

(d) *Stack Pointer* (43)

The *Stack Pointer* (43) designates lower 8 bits of the highest address on a stack region of the *RAM* (6), and is decremented after the pushing of data into the stack region and incremented before the pulling of the data from the stack region. For instance, 256 bytes of addresses *2100* to *21FF* are allocated to the stack region in a logical address.

(e) *source high register* (45), *destination high register* (46), and *length high register* (47)

These registers function in case of a command of a block transfer. The *source high register* (45) provides an upper byte of a source address to designate the source address together with a content of the x *register* (39). The *destination high register* (46) provides an upper byte of a destination address to designate the destination address together with a content of the *Y register* (40). The *length high register* (47) provides upper 8 bits for a down counter together with a content of the *Accumulator* (38) so that a length of a block transfer is counted by a byte unit.

(6) *Mapping Register* (25)

The *Mapping Register* (25) is composed of 8 registers each being of 8 bits to convert a logical address of 16 bits to a physical address of 21 bits, and is selected by upper 3 bits of the *H-bus* (35).

(7) *chip enable decoder* (26)

The *chip enable decoder* (26) provides chip enable outputs for the following peripheral circuits by decoding upper 11 bits of a physical address.

(a) a chip enable for the *RAM* (6) . . . $\overline{CER}$

(b) a chip enable for the *Video Display Controller* (2) . . . $\overline{CE7}$

(c) a chip enable for the *Video Color Encoder* (3) . . . $\overline{CEK}$

(d) a chip enable for the programable sound generator 4 . . . $\overline{CEP}$

(e) a chip enable for the *Timer* (29) . . . $\overline{CET}$

(f) a chip enable for the input and output port . . . $\overline{CEIO}$

(g) a chip enable for the *interrupt request register* (30) and the *interrupt disable register* (31) . . . $\overline{CECG}$

(8) *timing and control unit* (27)

The unit 27 is connected to following terminals.

(a) $\overline{RD}$ terminal

A read timing signal is supplied through the $\overline{RD}$ terminal at a reading cycle.

(b) $\overline{WR}$terminal

A write timing signal is supplied through the $\overline{WR}$ terminal at a writing cycle.

(c) SYNC terminal

A synchronous signal of *High* is supplied through the SYNC terminal at an instruction fetch cycle, that of *Low* is supplied therethrough at a system reset timing.

(d) $\overline{NMI}$ terminal

A non-maskable interrupt is produced when $\overline{NMI}$ input signal is supplied through the $\overline{NMI}$ terminal. A sub-routine call is conducted by reading lower address from the logical address *FFFC* and upper address from the logical address *FFFD* when a command which is conducted in a program is completed.

(e) $\overline{IRQ1}$ and $\overline{IRQ2}$ terminals

A sub-routine call is conducted by reading lower address from the logical address *FFF8* and upper address from the logical address *FFF9* when $\overline{IRQ1}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

A sub-routine call is conducted by reading lower address from the logical address *FFF6* and upper address from the logical address *FFF7* when $\overline{IRQ2}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

(f) $\overline{RESET}$ terminal

A program is started by reading lower address from the physical address *001FFE* and upper address from the physical address *001FFF* when a $\overline{RESET}$ input becomes *Low*.

(g) $\overline{RD}$Y terminal

The *CPU* (1) is started to operate when a $\overline{RD}$Y input is changed from *Low* to *High*.

(h) SX terminal

A complementary signal of a system clock signal is supplied through the SX terminal.

(i) OSC1 terminal

An external clock signal is input through the OSC1 terminal.

EAl to $\overline{EA3}$ terminals

These are input terminals for a test of the *CPU* (1).

(k) HSM terminal

A speed signal of *High* is supplied through the HSM terminal in case of a high speed mode of 21.47727 MHz/3, and that of *Low* is supplied therethrough in case of a low speed mode of 21.47727 MHz/12.

(9) *input and output port* (28)

The *input and output port* (28) is connected to following terminals.

(a) K0 to K7 terminals

The terminals are input ports from which data are written in accordance with the conduct of a reading cycle in regard to the physical addresses *1FF000* to *1FF3FF*.

(b) 00 to 07 terminals

The terminals are output ports with latches to which data are supplied in accordance with the conduct of a writing cycle in regard to the physical addresses *1FF000* to *1FF3FF*.

(10) *Timer* (29)

The *Timer* (29) is connected to a test input terminal EAT for the *CPU* (1) and provides a timer signal through the U-bus thereto.

(11) *interrupt request register* (30)

The *register* (30) is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *1* to show the IRQ1 and IRQ2 terminals *Low* and the remaining 1 bit is *1* to show a timer interrupt caused. The *register* (30) is only used for *read*.

(12) *interrupt disable register* (31)

The *register* (31) is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *0* to show an interrupt request of the IRQ1 and IRQ2 terminals disable, and the remaining one is *0* to show an interrupt request disable in accordance with the timer interrupt.

In [Figure 3], there is shown the aforementioned *Mapping Register* (25) comprising 8 registers MPR0 to MPR7 each being of 8 bits, and connected through the *B-bus* (32) to an *input output controller* (50) and through an *output selector* (51) to the output terminals A13 to A20. The *output selector* (51) selects one register from the 8 registers MPR0 to MPR7 of the *Mapping Register* (25) in accordance with upper 3 bits H5 to H7 of upper data of a logical address on the *H-bus* (35).

File:US5566313-fig.png
[Figure 3] is a block diagram showing a mapping register circuit in an apparatus for controlling the transfer of data in the embodiment.

[Figure 4] shows a relation between the upper 3 bits H5 to H7 and one register selected from the 8 registers MPR0 to MPR7. If it is assumed that the upper 3 bits H5 to H7 are *010*, the register MPR2 is selected from the 8 registers MPR0 to MPR7. Data are read from the *Mapping Register* (25) by a command *TMAi* where *i* is an integer selected from *0* to *7*. For instance, data are read from the register MPR2 to be transferred through the *B-bus* (32) to the *Accumulator* (38) in accordance with a command TMA_2. On the other hand, data are written into the *Mapping Register* (25) by a command *TAMi* where *i* is an integer selected from *0* to *7*. For instance, data are transferred to be written into the register MPR0 from the *Accumulator* (38) by a command TAM_0. The commands *TMAi* and *TAMi* are composed of two byte respectively, and lower byte thereof includes a bit of *1* corresponding in a bit number to a register number which is selected from the 8 registers MPR0 to MPR7 and remaining 7 bits of *0*. When one of the 8 registers MPR0 to MPR7 is selected in accordance with upper 3 bits of the *H-bus* (35), a content of the selected register is supplied through the output terminals A13 to A20 to a following stage so that a physical address of 21 bit is obtained together with a content of the *logical address low register* (48) to be supplied through the output terminals A0 to A7 thereto, and lower 5 bits of the *logical address high register* (49) to be supplied through the output terminals A8 to A12 thereto. When the most significant bit A20 of a physical address A0 to A20 is *1*, a command by which data designated in accordance with the physical address A0 to A20 are immediate-transferred to the *Video Display Controller* (2) is conducted. The command includes ST0, ST1 and ST2 by which codes are produced on A0 and A1 of the address bus at a write cycle as shown in [Figure 6]. The command is set in the *instruction register* (20).

File:US5566313-fig.png
[Figure 4] is an explanatory diagram showing a mapping register selection code when a physical address is produced in an apparatus for controlling the transfer of data in the embodiment.

In operation, lower data of a logical address are set in the *logical address low register* (48), and lower 5 bits of upper data of the logical address are set in the *logical address high register* (49). Here, it is assumed that address data are set in the 8 registers MPR0 to MPR7 of the *Mapping Register* (25) respectively. When upper 3 bits of upper data of the logical address, that is to say, upper 3 bits H5 to H7 of the *H-bus* (35) are *001*, the register MPR1 is selected so that a content *F8* of the register MPR1 is supplied through the output terminals A13 to A20 to the following stage. These output signals are combined with output signals of the output terminals A8 to A12 and A0 to A7 to produce a physical address A0 to A20. Then, the *chip enable decoder* (26) decodes upper 11 bits A10 to A20 of the physical address A0 to A20 to produce a chip enable signal $\overline{CER}$ of *0* by which a data memory is enabled.

When the immediate data transfer command is produced, the signal $\overline{CE7}$ is *0*, the MSB A20 is *1*, and the bits A0 and A1 are of a content as shown in [Figure 6] so that data are transferred from the *CPU* (1) to the *Video Display Controller* (2) as shown in detail in [Figure 7].

[Figure 7] shows the *Video Display Controller* (2) which comprises a *control unit* (70), an *address unit* (71) a *CPU read/write buffer* (72), a *Sprite Attribute Table buffer* (73), a sprite attribute shift *register* (74), a *background shift register* (75), a *data bus buffer* (76), a *synchronous circuit* (77), and a *priority circuit* (78).

File:US5566313-fig.png
[Figure 7] is a block diagram showing the video display controller in the apparatus for displaying a color image.

The *control unit* (70) is provided with a $\overline{BUSY}$ terminal through which *0* is supplied to the *CPU* (1) in a case where the *Video Display Controller* (2) is not in time for the writing/reading of data from the *CPU* (1) to the *VRAM* (7) so that the *CPU* (1) is held under the state, a $\overline{IRQ}$ terminal through which an interrupt request signal is supplied to an external circuit, a CK terminal through which a clock signals having a frequency for one pixel (one picture element) is received, a $\overline{RESET}$ terminal through which a reset signal for initiallization is received, and a EX8/16 terminal through which a data bus change-over signal for changing over a data bus width is received.

The *address unit* (71) is connected to address terminals MA0 to MA15 through which an address signal of the VRAM is supplied thereto. An address region of the *VRAM* (7) is of a capacity, for instance, 65.536 words, provided that one word is of 16 bits. The *address unit* (71), the *CPU read/write buffer* (72), the *Sprite Attribute Table* (73), the *sprite shift register* (74) and the *background shift register* (75) are connected through data bus to data terminals MD0 to MD15 through which data of the *VRAM* (7) are transferred for the writing thereinto or the reading therefrom.

The *Sprite Attribute Table buffer* (73) is a memory for storing a display position (X, Y), color, pattern number and so on of a sprite (16 bits).

The *sprite shift register* (74) is supplied with a pattern number, sprite color etc. read from the *Sprite Attribute Table buffer* (73) to access the *VRAM* (7), and stores data for a pattern, color etc. of a sprite read from a sprite generator in the *VRAM* (7).

The *background shift register* (75) stores a pattern and CG color of a background. The pattern is read from a character generator in accordance with an address of the character generator in the *VRAM* (7) obtained from a character code which is read from an attribute table in the *VRAM* (7) together with the CG color.

The *data bus buffer* (76) is connected to terminals D0 to D15 through which data are supplied to and received from an external circuit. The *Video Display Controller* (2) can select one of 8 bit interface and 16 bit interface in compliance with a data width of a system including the *CPU* (1). When the 8 bit interface is selected, the terminals D0 to D7 are used among the

terminals D0 to D15.

The *synchronous circuit* (77) is connected to a DISP terminal through which a signal for indicating a display period is supplied to an external circuit, a $\overline{\text{VSYNC}}$ terminal through which a signal for a vertical synchronization of a CRT is supplied to the *video display* (9) and an external vertical synchronous signal is received, and a $\overline{\text{HSYNC}}$ terminal through which a signal for a horizontal synchronization of the CRT is supplied to the *video display* (9) and an external horizontal synchronous signal is received.

The *priority circuit* (78) is connected to terminals VD0 to VD7 through which video data are supplied to an external circuit, and a *SP/BG* (VDB) terminal through which a signal *1* is supplied to an external circuit when the video data are for a sprite and a signal *0* is supplied thereto when the video data are for a background.

The aforementioned *control unit* (70) is further provided with a $\overline{\text{CS}}$ terminal through which a signal *0* is received so that registers therein are accessed for the writing and reading of data from the *CPU* (1), $\overline{\text{RD}}$ and $\overline{\text{WR}}$ terminals through which read and write-timing signals are received, and terminals A0 and A1 connected to the address bus of the *CPU* (1) as shown in [Figure 2].

The *Video Display Controller* (2) is further provided with a $\overline{\text{MRD}}$ terminal and a $\overline{\text{MWR}}$ terminal. When the $\overline{\text{MRD}}$ terminal is under a state of *0*, data are read from the *VRAM* (7) to the *CPU* (1). On the other hand when the $\overline{\text{MWR}}$ terminal is under a state of *0*, data are written into the *VRAM* (7) from the *CPU* (1).

The *control unit* (70) includes a memory address write register and a VRAM data write register together with other registers.

In operation, when an immediate data transfer command (STO, ST1 or ST2) is produced, data transferred from the *CPU* (1) are stored in the VRAM data write register. The aforementioned physical address A0 to A20 is automatically set in the memory address write register without receiving an address setting command. As a result, the data are written into the *VRAM* (7) in synchronization with a write signal of *0* applied to the $\overline{\text{WR}}$ terminal in accordance with a single transfer command which is called "an immediate data transfer command".

That is to say, a chip including the aforementioned memory address write register and VRAM data write register is enabled in accordance with a chip enable signal $\overline{\text{CE7}}$ which becomes *0* in the conduct of the immediate data transfer command of ST0, ST1 or ST2. As a result, bits A0 and A1 of a physical address A0 to A20 of 21 bits becomes a bit state dependent on the command (STO, St1 or ST2). Accordingly, the data transferred from the *CPU* (1) are automatically stored in the *VRAM* (7) without the necessity that an address is set in the memory address write register by a command.

[Figure 8] shows a relation between a logical address of the *CPU* (1) and a physical address of the *RAM* (6) wherein a block *F8* of a physical address region having 2M bytes is allocated in accordance with a content *F8* of the selected register MPR1 in a case where a logical address of 16 bits on the Land H-buses 34 and 35 corresponds to one address selected from addresses *2000* to *3FFF* of a logical address region having 64K bytes as indicated by hatching lines therein. Thus, a memory of 2M bytes in which an address signal of 21 bits is required for the access thereof can be accessed by an address signal of 16 bits. At this time, data at a corresponding address of a memory are immediately transferred to a *Video Display Controller* (2) in a case where lower 2 bits A0 and A1 of the physical address are of a content as shown in [Figure 6].

File:US5566313-fig.png
[Figure 8] is an explanatory diagram showing a relation between logical and physical address regions in an apparatus for controlling the transfer of data in the embodiment.

[Figure 9] shows a physical address region of the *RAM* (6) to which a chip enable signal $\overline{\text{CER}}$ is allocated. The chip enable signal $\overline{\text{CER}}$ is produced in accordance with a content of upper 11 bits A10 to A20 of a physical address A0 to A20 which is decoded in the *chip enable decoder* (26). For instance, the chip enable signal $\overline{\text{CER}}$ is allocated to a region of physical addresses *1F0000* to *1F7FFF* in which zero page and stack regions exist in a case where a content of the selected register MPR1 is *F8*. The zero page and stack regions are of a region to which contents of the *Accumulator* (38), the *X register* (39), and the *Y register* (40) are evacuated temporarily.

File:US5566313-fig.png
[Figure 9] is an explanatory diagram showing an allocation of a chip enable signal to a physical address region in an apparatus for controlling the transfer of data in the embodiment.

Although the invention has been described with respect to specific embodiment for complete and clear disclosure, the appended claims are not to thus limited but are to be construed as embodying all modification and alternative constructions that may occur to one skilled in the art which fairly fall within the basic teaching herein set forth.

Patent for VRAM access via VDC/CPU

Note: terminals with a *High* value are a binary *1* and *Low* is a binary *0*.

Full Patent

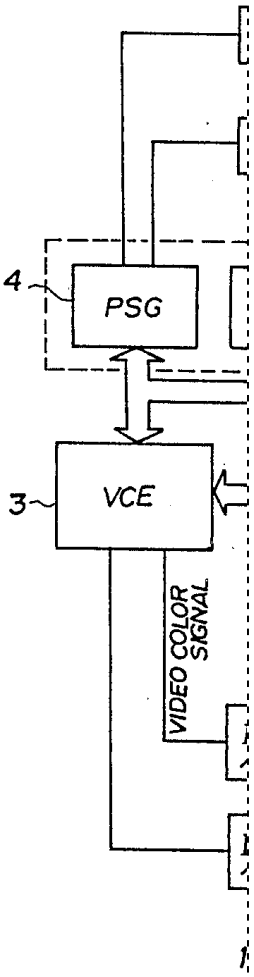
Original Patent (http://www.google.com/patents/US5030946)

United States Patent Number: **5030946**

APPARATUS FOR THE CONTROL OF AN ACCESS TO A VIDEO MEMORY

In [Figure 1], there is shown an apparatus for displaying an image on a screen which is mainly composed of a *Video Display Controller* (1), a *CPU* (2), a *Video Color Encoder* (3), and a *Programmable Sound Generator* (4). The *Video Display Controller* (1) supplies the *Video Color Encoder* (3) with image data for a story which are read from a *VRAM* (7) under the control of the *CPU* (2) reading

a program stored in a *ROM* (5). The *CPU* (2) controls a *RAM* (6) to store data, calculation or arithmetical results etc. temporarily in accordance with a program stored in the *ROM* (5). The *Video Color Encoder* (3) is supplied with image data to produce RGB analog signals or video color signals including luminance signals and color difference signals to which the RGB signals are matrix-converted by using color data stored therein. The programmable sound generator 4 is controlled by the *CPU* (2) reading a program stored in the *ROM* (5) to produce audio signals making left and right stereo sounds. The video color signals produced at the *Video Color Encoder* (3) are of composite signals supplied through an *interface* (8) to a *video display* (9), while the RGB analog signals are directly supplied to a *video display* (9) which is used as an exclusive monitor apparatus. The left and right analog signals supplied from the *Programmable Sound Generator* (4) are amplified at amplifiers 11a and 11b to make sounds at speakers 12a and 12b.



[Figure 1] is a block diagram in which an apparatus for the invention is included.

In [Figure 2A], there is shown the *Video Display Controller* (1) transferring data between the *CPU* (2) and *VRAM* (7) which comprises a *control unit* (20) including various kinds of registers to be described later, an *address unit* (21), a *CPU read/write buffer* (22), and *sprite shift register* (24), a *background shift register* (25), a *data bus buffer* (26), a *synchronic circuit* (27), and a *priority circuit* (28).

The *control unit* (20) is provided with a $\overline{\text{BUSY}}$ terminal being *Low* to keep the *CPU* (2) writing data into the *VRAM* (7) or reading data therefrom in a case where the *Video Display Controller* (1) is not in time for the writing or reading of the date, an $\overline{\text{IRQ}}$ terminal supplying an interruption request signal, a CK terminal receiving a clock signal of a frequency for one pixel periods (one picture element), a $\overline{\text{RESET}}$ terminal receiving a reset signal for initializing the *Video Display Controller* (1), and an EX 8/$\overline{16}$ terminal receiving a data bus width signal for selecting one of 8 and 16 bit data buses.

The *address unit* (21) is connected to terminals MAO to MA15 supplying address signals for the *VRAM* (7) which has, for instance, a special address region of 65,536 words. The *address unit* (21), *CPU read/write buffer* (22), *Sprite Attribute Table* (23), *sprite shift register* (24), and *background shift register* (25) are connected to terminals MD 0 to MD 15 through which data are transferred to and from the *VRAM* (7).

The *Sprite Attribute Table buffer* (23) is a memory for storing X and Y display positions, pattern codes and control data of sprites each composed of 16×16 pixels as described in more detail later.

The *sprite shift register* (24) stores pattern and color data of a sprite read from a sprite generator in the *VRAM* (7) which is accessed in accordance with the pattern codes stored in the *Sprite Attribute Table* (23) as described in more detail later.

The *background shift register* (25) stores pattern data, along with CG color, read from a character generator in the *VRAM* (7) in accordance with an address based on a character code of a background attribute table in the *VRAM* (7) which is accessed in an address decided by a raster position as also described in more detail later.

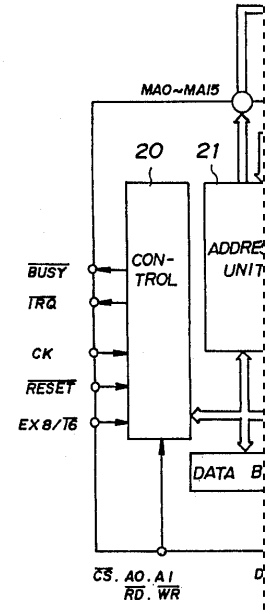
The *data bus buffer* (26) is connected to terminals D0 to D15 through which data are supplied and received. In the *Video Display Controller* (1), 8 or 16 bit interface is selected to comply

with a data width of a system including the CPU2 wherein the terminals D0 to D7 among the terminals D0 to D15 are occupied when the 8 bit interface is selected.

The *synchronic circuit* (27) is connected to a DISP terminal indicating a display period, a $\overline{\text{VSYNC}}$ terminal from which a vertical synchronous signal for a *video display* (9) is supplied and in which an external vertical synchronous signal is received, and a $\overline{\text{HSYNC}}$ terminal from which a horizontal synchronous signal for a *video display* (9) is supplied and in which an external horizontal synchronous signal is received.

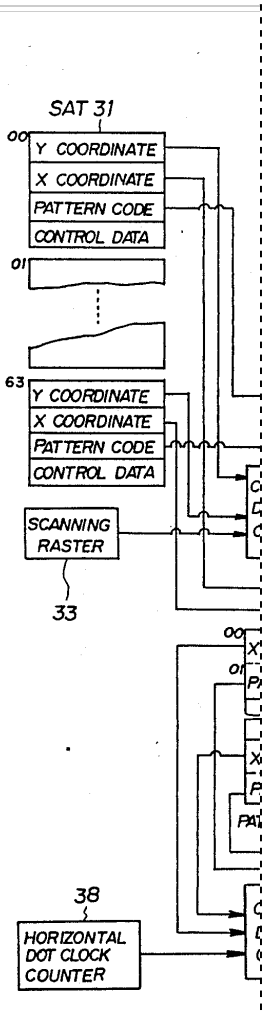
The *priority circuit* (28) is connected to terminals VD0 to VD7 through which video signals are supplied, and a *SP/BG* (VD8) terminal being *High* when the video signals are of a sprite and being *Low* when the video signals are of a background.

The aforementioned *control unit* (20) is also connected to a $\overline{\text{CS}}$ terminal being *Low* wherein the *CPU* (2) is able to read data from registers therein and sprite data thereinto, a $\overline{\text{RD}}$ terminal receiving a clock signal for the reading thereof, a $\overline{\text{WR}}$ terminal receiving a clock signal for the writing thereof, and terminals A0 and A1 which are connected to address bus of the *CPU* (2). Further, the *Video Display Controller* (1) is provided with a $\overline{\text{MRD}}$ terminal being *Low* when the *CPU* (2) reads data from the *VRAM* (7), and a $\overline{\text{MWR}}$ terminal being *Low* when the *CPU* (2) writes data into the *VRAM* (7).



[Figure 2] is a block diagram writing video signals into a

In [Figure 2B], there is shown an apparatus for displaying a sprite on a screen which is included in the apparatus of [Figure 1] wherein the reference numerals 31 and 32 indicate a sprite attribute table and sprite generator in the *VRAM* (7) respectively. The *Sprite Attribute Table* (31) can include, for instance, sixty-four sprites, while the *Sprite Generator* (32) can include, for instance, one thousand and twenty-four sprites. In the *Sprite Attribute Table* (31), addresses of 0 to 63 are assigned to the sixty-four sprites to give a priority thereto in the order of the address 0>1>. . . >62>63. Each of the sprites is composed of 16×16 bits, and includes X and Y coordinates, pattern codes and control data. As to each of the sprites, the Y coordinate is compared with a raster signal supplied from a scanning raster producing circuit 33 in a *coincidence detection circuit* (34) whereby sprites each having a Y coordinate coincident with a raster signal are stored into a *pattern code buffer* (35) which can store a maximum number of 16 sprites by referring to a corresponding one of the addresses 0 to 63. A *selector* (36) selects a pattern code of the *Sprite Attribute Table* (31) in accordance with an address stored in the *pattern code buffer* (35) to access the *Sprite Generator* (32) in regard to an address which is of a selected pattern code, thereby reading pattern data from the *Sprite Generator* (32). The pattern data thus obtained are stored into a *pattern data buffer* (37) along with an X coordinate corresponding thereto read from the *Sprite Attribute Table* (31). The storing of sprites into the *pattern code buffer* (35) is performed at a horizontal display period preceding to the present horizontal display period by one scanning raster, while the storing of pattern data into the *pattern data buffer* (37) is performed at a following horizontal retrace period. When a scanning raster at which pattern data are displayed has come, the X coordinate thus stored in the *pattern data buffer* (37) is compared with a counted value of a horizontal pixel period clock counter 38 in a *coincidence detection circuit* (39) whereby pattern data having an X coordinate coincident with the counted value are supplied to a *parallel/serial converting circuit* (40). In the *parallel/serial converting circuit* (40), parallel pattern data are converted into serial pattern data which are supplied through a *gate circuit* (42) to a *video display* (9). The *gate circuit* (42) is controlled to be turned on and off in accordance with a content of a *starting coordinates registration circuit* (43) by the *CPU* (2). The content thereof is X and Y coordinates by which the starting coordinates of a display region is defined on a display screen.



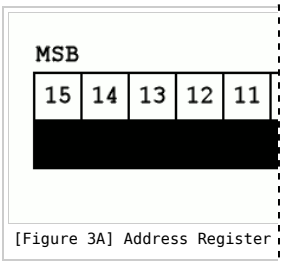
[Figure 2] is a block diagram in the apparatus of [Figure 1

In [Figures 3A to 3U], there are shown various kinds of registers included in the Control Unit (20) of the Video Display Controller (1).

(a) Address Register ([Figure 3A])
A register number *AR* is exclusive written into the address register designating one of the memory address write register to DMA VRAM-SATB source address register as shown in [Figures 3C to 3U] so that data are writing into the *Video Display Controller* (1) under the condition that the A1 and $\overline{CS}$ terminals thereof are *Low*.

In a case where 16 bit data bus is selected, the EX 8/$\overline{16}$ terminal is *0*, the A1 terminal is *0*, the *R/W* terminal is *W*, and the A0 terminal is no matter.

In a case where 8 bit data bus is selected, the EX 8/$\overline{16}$ terminal is *1*, the A0 and A1 terminals are *0*, and the *R/W* terminal is *W*.



[Figure 3A] Address Register

(b) Status Register ([Figure 3B])
A bit corresponding to one of interruption jobs is set to be *High* in the status register to make the interruption active when a cause of the interruption which is enabled by an interruption permission bit of a *Control Register* and *DMA Control Register* as showing in [Figures 3G and 3Q] is occurred. When the status is read from the status register, the corresponding bit is cleared automatically.

The status indicating bits are as follows.

(1) bit 0 (CR) - collision of sprites
It is indicated that the sprite number 0 of a sprite is collided with any one of the sprite numbers 1 to 63 of sprites.

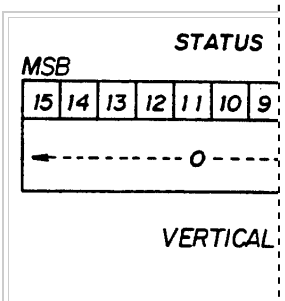
(2) bit 1 (OR) - more sprites than a predetermined number
(2.1) a case where more than 17 sprites are detected on a single raster line.
(2.2) a case where data of a sprites which is designated are not transferred to a data buffer in a horizontal trance period.
(2.3) a case where a bit of CGX in control data of a sprite by which two sprites are joined in a horizontal direction is set so that data of the sprites are not transferred to a data buffer.

(3) bit 2 (PR) - detection of raster
It is indicated that a value of a raster counter becomes a predetermined value of a raster detecting register.

(4) bit 4 (DS) - finishing of DMA transfer
It is indicated that data transfer between the *VRAM* (7) and *Sprite Attribute Table buffer* (23) is finished.

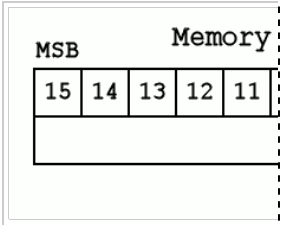
(5) bit 4 (DV) - finishing of DMA transfer
It is indicated between two regions of *VRAM* (7) is finished.

(6) bit 5 (VD) - vertical retrace period
It is indicated that the *VRAM* (7) accessed for the writing or reading of data by the *CPU* (2) so that the $\overline{BUSY}$ terminals is *0*.



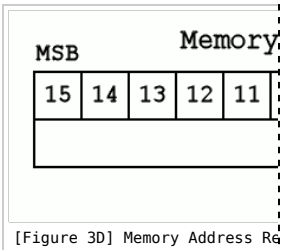
[Figure 3B] Status Register (

(c) Memory Address Write Register (register name *0x00*, [Figure 3C])
A starting address *MAWR* is written into the memory address write register so that the writing of data begins at the starting address of the *VRAM* (7).



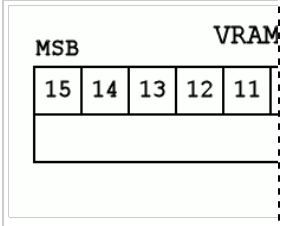
[Figure 3C] Memory Address Wr

(d) Memory Address Read Register (register number *0x01*, [Figure 3D])
A starting address *MARR* is written into the memory address read register. When the upper byte of the starting address is written thereinto, data are begun to be read from the starting address of the *VRAM* (7) so that data thus read are written into a VRAM data read register as showing in [Figure 3F]. There after, the starting address *MARR* is automatically incremented by one.



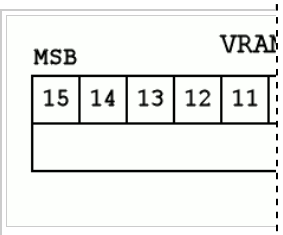
[Figure 3D] Memory Address Re

(e) VRAM Data Write Register (register number *0x02*, [Figure 3E])
Data which are transferred from the *CPU* (2) to the *VRAM* (7) are written into the VRAM data write register. When the upper byte of the data *VWR* is written thereinto, the *Video Display Controller* (1) begins to write the data into the *VRAM* (7) and the address *MAWR* of the memory address write register is automatically incremented by one upon writing of the data.



[Figure 3E] VRAM Data Write R

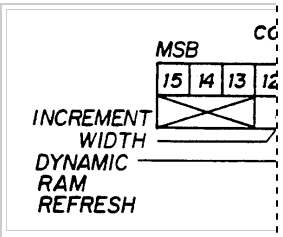
(f) VRAM Data Read Register (register number *0x02*, [Figure 3F])
Data which are transferred from the *VRAM* (7) to the *CPU* (2) are written into the VRAM data read register. When the upper byte of the data *VRR* is read therefrom, the reading of data is performed at the following address of the *VRAM* (7).



[Figure 3F] VRAM Data Read Re

(g) Control Register (register number *0x05*, [Figure 3G])
An operating mode of the *Video Display Controller* (1) is controlled in accordance with the following bits of the *Control Register*.

(1) bits 0 to 3 (IE) - enable of interruption request
(1.1) bit 0 - collision detection of sprites
(1.2) bit 1 - excess number detection of sprites
(1.3) bit 2 - raster detection
(1.4) bit 3 - detection of vertical retrace period



[Figure 3G] Control Register

(2) bits 4 and 5 (EX) - external synchronization

bit 5	bit 4	content
0	0	$\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ are inputs, and synchronous to external signals
0	1	$\overline{\text{VSYNC}}$ is and input, and sychronous to external signals, while $\overline{\text{HSYNC}}$ is an output
1	0	non-used
1	1	$\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ are outputs

(3) bit 6 (SB) - sprite blanking
It is decided whether a sprite should be displayed on a screen or not.
The control of the bit is effective in the following *Horizontal Display Period*.
(3.1) *0* - blanking of a sprite
(3.2) *1* - display of a sprite

(4) bit 7 (BB) - background blanking
It is decided whether background should be displayed on a screen or not.
The control of the bit is effective in the following *Horizontal Display Period*.
(4.1) *0* - blanking of background
(4.2) *1* - display of background.

(3.4) As a result, when bits 6 and 7 are both *0*, there is a resulted "burst mode" in which the following operations can be performed.
(3.4.1) The access to the *VRAM* (7) is not performed for a display, but the *VRAM* (7) is accessed by the *CPU* (2).
(3.4.2) DMA between two regions of the *VRAM* (7) is possible to be performed at any time.

In such occasions, the terminals VD0 and VD7 are all *Low* while the *SP/BG* terminal is *High*. On the other hand, when the bits 6 and 7 are both *1*, there is released from the "burst mode".

(5) bits 8 and 9 (TE) - selection of DISP terminal outputs

Bit 9	Bit 8	DISP output	Content
0	0	DISP	output "H" during display
0	1	BURST	color burst inserting position is indicated by output "L"
1	0	INTHSYNC	internal horizontal synchronous signal
1	1		non-used

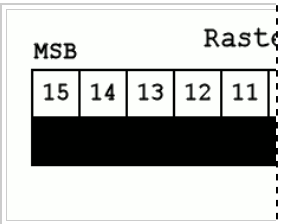
(6) bit 10 (DR) - dynamic RAM refresh
Refresh address is supplied from the terminals MA0 to MA15 upon the setting of the bit in a case where VRAM pixel width is of 2 pixels or 4 pixels for background in a memory width register as showing in [Figure 3K].

(7) bits 11 and 12 (IW) - increment width selection of memory address write register or memory address read register
A width which is incremented in address is selected as follows.

bit 12	bit 11	increment width
0	0	+1
0	1	+20H
1	0	+40H
1	1	+80H

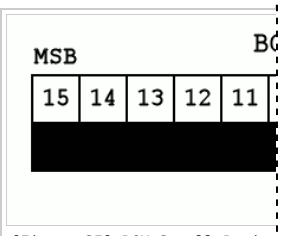
In a case of 8 bit access, an address is incremented upon the upper byte.

(h) Raster Detecting Register (register number *0x06*, [Figure 3H])
A raster number *RCR* at which an interruption job is performed is written into the raster detecting register. An interruption signal is produced when a value of a raster counter is equal to the raster number *RCR*. The raster counter is preset to be *64* at a preceding scanning raster line to a display starting raster line as described in more detail later, and is increased at each raster line by one.



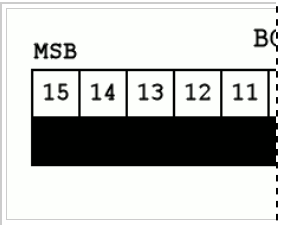
[Figure 3H] Raster Detecting

(i) BGX Scroll Register (register number *0x07*, [Figure 3I])
The BGX scroll register is used for a horizontal scroll of background on a screen. When a content *BXR* is rewritten therein, the content is effective in the following raster line.



[Figure 3I] BGX Scroll Regist

(j) BGY Scroll Register (register number *0x08*, [Figure 3J])
The BGY scroll register is used for a vertical scroll of background on a screen. When a content *BYR* is rewritten therein, the content is effective to be as "BYR + 1" in the following raster line.

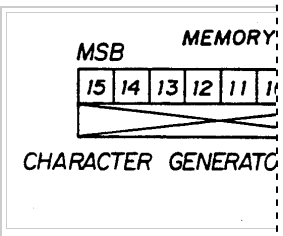


[Figure 3J] BGY Scroll Regist

(k) Memory Width Register (register number *0x09*, [Figure 3K])
(1) bits 0 and 1 (VM) - VRAM pixel width
A pixel width in which an access to the *Background Attribute Table* and character generator, DMA and access of the *CPU* (2) to the *VRAM* (7) during a horizontal display period are performed is written into the bits of the memory width register. The pixel width is decided dependent on a memory speed of the *VRAM* (7). When the bits 0 and 1 are re-written therein, the content is effective at the beginning of a vertical retrace period.

bit 1	bit 0	dot width	Disposition in one character cycle (8 dots) 1	2	3	4	5	6	7	8
0	0	1	CPU	BAT	CPU		CPU	CG0	CPU	CG1
0	1	2	BAT		CPU		CG0		CG1	
1	0	2	BAT		CPU		CG0		CG1	
1	1	4	BAT				CG0/CG1			

BAT is for *Background Attribute Table*, and *CG* is for character generator.



[Figure 3K] Memory Width Regi

(2) bits 2 and 3 (SM) - sprite pixel width
A pixel width which an access to the sprite generator is performed during a horizontal retrace period is written into the bits of the memory width register.

bit 3	bit 2	dot width	Disposition in one character cycle (8 dots) 1	2	3	4	5	6	7	8
0	0	1	SP0	SP1	SP2	SP3	SP0	SP1	SP2	SP3
*0	1	2	SP0		SP1		SP0		SP1	
			SP2		SP3		SP2		SP3	
1	0	2	SP0		SP1		SP2		SP3	
**1	1	4		SP0				SP1		
				SP2				SP3		

(note)
*(SP0 SP1) or (SP2 SP3) is selected dependent on LSB bit of a pattern code.
**SP0 to SP3 are read in two consecutive character cycles.

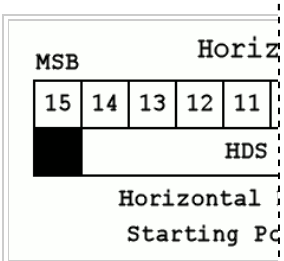
(3) bits 4 to 6 (SCREEN)
The number of character in X and Y directions of a fictitious screen is decided dependent on the content of the bits. When a content is effective at the beginning of a vertical retrace period.

bit 6	bit 5	bit 4	Number of characters X	Y
0	0	0	32	32
0	0	1	64	32
0	1	0	128	32
0	1	1	128	32
1	0	0	32	64
1	0	1	64	64
1	1	0	128	64
1	1	1	128	64

(4) bit 4 (CM) - CG mode
When a VRAM pixel width is of 4 pixels, a color block of a character generator is changed dependent on the bit. A content is writtent into the bit, the content is effective in the following raster line.

(l) Horizontal Synchronous Register (register number *0x0A*, [Figure 3L])
(1) bits 1 to 4 (HSW) - horizontal synchronous pulse
A pulse width of *Low* level of a horizontal synchronous pulse is set as an unit of a character cycle. One of 1 to 32 is selected by using 5 bits to comply with the specification of a *video display*.

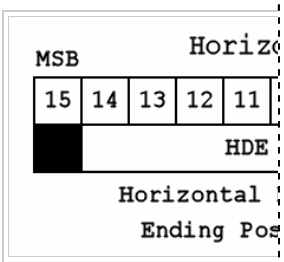
(2) bits 8 to 14 (HDS) - starting position of horizontal display
A period between a rising edge of a character cycle. An optimum position in a horizontal direction on a *video display* is decided by a content of the 7 bits. When it is assumed that a horizontal display position (horizontal back porch) is *N*, *N - 1* is written into HDS bits.



[Figure 3L] Horizontal Synchr

(m) Horizontal Display Register (register number *0x0B*, [Figure 3M])
(1) bits 0 to 6 (HDW) - horizontal display width
A display period in each raster line is set as an unit of a character cycle, and is decided in accordance with the number of characters in the horizontal direction on a *video display* dependent on a content of the 7 bits. If it is assumed that a horizontal display position is *N*, *N - 1* is written into HDW bits.

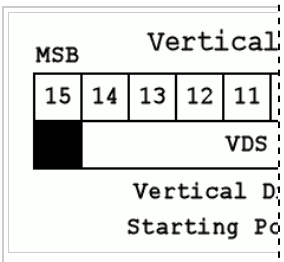
(2) bits 8 to 11 (HDE) - horizontal display ending position
A period between an ending of a *Horizontal Display Period* and a rising edge of a horizontal synchronous signal is set as an unit of a character cycle. An optimum position of a horizontal display is set on a *video display* by the 7 bits. When it is assumed that a horizontal display ending position (horizontal back porch) is *N*, *N -1* is written into HDE bits.



[Figure 3M] Horizontal Displa

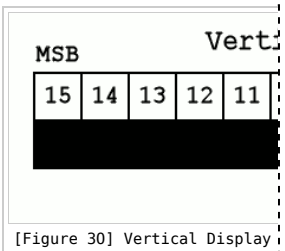
(n) Vertical Synchronous Register (register number 0x0C, [Figure 3N])
(1) bits 0 to 4 (VSW) - vertical synchronous pulse width
A pulse width of a vertical synchronous signal is decided in a width of *Low* level as a unit of a raster line. One of 1 to 32 is selected to comply with a specification of a *video display*.

(2) bits 8 to 15 (VDS) - vertical display starting position
A period between a rising edge of a vertical synchronous signal and a vertical synchronous starting position is set as an unit of a raster line. When it is assumed that a vertical display starting position (vertical back porch) is *N*, *N-2* is written into the bits.



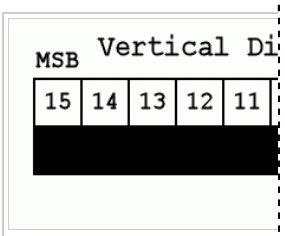
[Figure 3N] Vertical Synchron

(o) Vertical Display Register (register number *0x0D*, [Figure 3O])
A vertical display period (display region) is set as an unit of a raster line. A vertical display width is decided in accordance with the number of raster lines to be displayed on a *video display* which is defined by a content of the 9 bits. When it is assumed that a vertical display width is *N*, *N - 1* is written into the VDW bits.



[Figure 3O] Vertical Display

(p) Vertical Display Ending Position Register (register number *0x0E*, [Figure 3P])
A period between a vertical display ending position and a rising edge of a vertical synchronous signal is set as an unit of a raster line. When it is assumed that a vertical optimum position (vertical front porch) is *N* to be defined by the 8 bits, *N* is written into the VCR bits.



[Figure 3P] Vertical Display

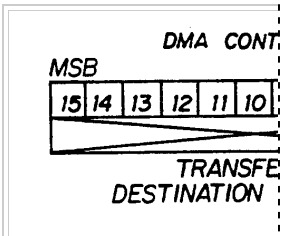
(q) DMA Control Register (register number *0x0F*, [Figure 3Q])
(1) bit 0 (DSC) - Enable an interruption at the finishing of transfer between the *VRAM* (7) and *Sprite Attribute Table buffer* (23).
It is decided whether or not an interruption is enabled at the finishing time of the transfer.
(1.1) *0* - disable
(1.2) *1* - enabled

(2) bit 1 (DVC) - Enable an interruption at the finishing of transfer between two regions of the *VRAM* (7).
It is decided whether or not an interruption is enabled finishing time of the transfer.
(2.1) *0* - disable
(2.2) *1* - enabled

(3) bit 2 (SI/D) - Increment/decrement of a source address
One of automatically increment and decrement of a source address is selected in a transfer between two regions of *VRAM* (7).
(3.1) *0* - increment
(3.2) *1* - decrement

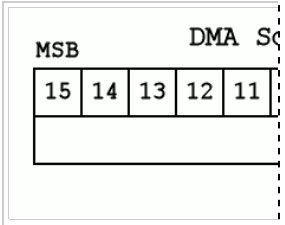
(4) bit 3 (DI/D) - Increment/decrement of a destination address
One of automatically increment and decrement of a destination address is selected in a transfer between two regions of *VRAM* (7).
(4.1) *0* - increment
(4.2) *1* - decrement

(5) bit 4 (DSR) - repetition of a transfer between the *VRAM* (7) and the *Sprite Attribute Table buffer* (23) is enabled.
(5.1) *0* - disable
(5.2) *1* - enabled



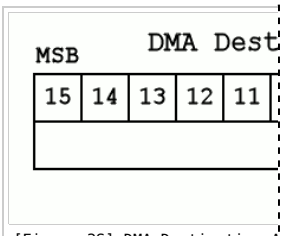
[Figure 3Q] *DMA Control Regis*

(r) DMA Source Address Register (register number *0x10*, [Figure 3R])
A starting address of a source address is allocated in a transfer between two regions of the *VRAM* (7).



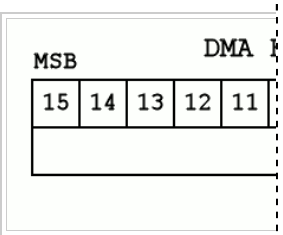
[Figure 3R] DMA Source Addres

(s) DMA Destination Address Register (register number *0x11*, [Figure 3S])
A starting address of a destination address is allocated in a transfer between two regions of the *VRAM* (7).



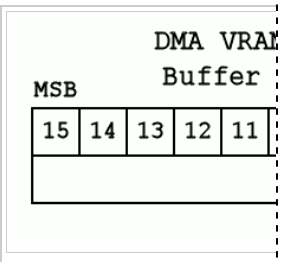
[Figure 3S] DMA Destination A

(t) DMA Block Length Register (register number *0x12*, [Figure 3T])
A length of a block is defined in a transfer between two regions of the *VRAM* (7).



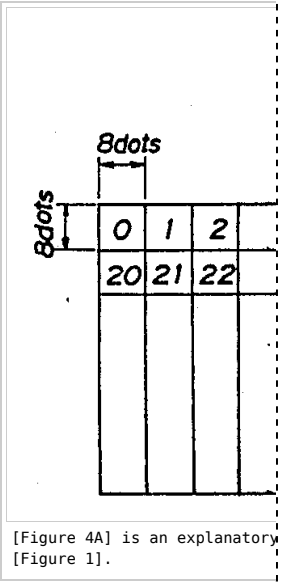
[Figure 3T] DMA Block Length

(u) DMA VRAM-SATB Source Address Register (register number *0x13*, [Figure 3U])
A starting address of a source address is allocated in a transfer between the *VRAM* (7) and *Sprite Attribute Table buffer* (23).



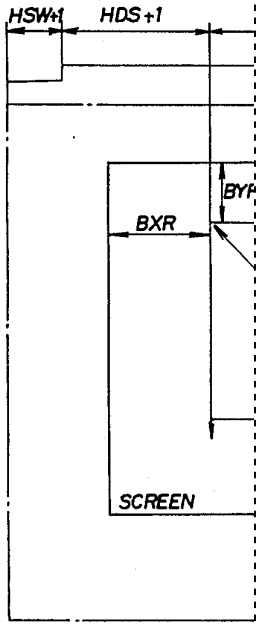
[Figure 3U] DMA VRAM-SAT Sour

In [Figure 4A, there is shown an address in a background attribute table for a character on a fictitious screen. A character and color to be displayed at each character position are stored in the background attribute table. A predetermined number of background attribute tables are stored in a region the first address of which is *0* in the *VRAM* (7). The fictitious screen shown therein which is one example is of 32×32 characters (1F=32).



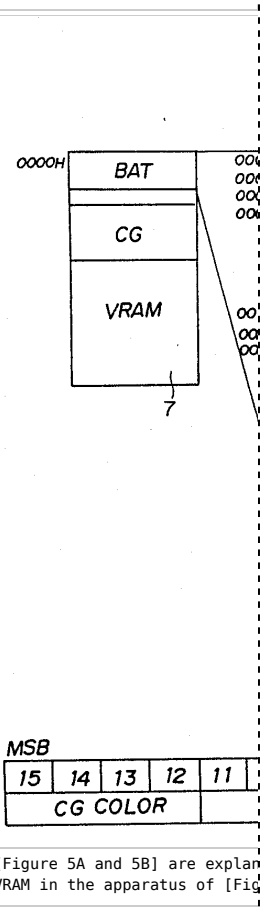
[Figure 4A] is an explanatory [Figure 1].

In [Figure 4B, there is shown a screen which is framed by writing respective predetermined values into the aforementioned horizontal synchronous register, horizontal display register, vertical synchronous register and vertical display register as shown in [Figures 3L, 3M, 3N and 3O]. Although the respective predetermined values for the registers are not explained here, a display region is defined in accordance with "HDW+1" in the horizontal display register and "VDW+1" in the vertical display register. In the embodiment, the starting coordinates (x,y) for the display region is indicated to be as (32, 64).



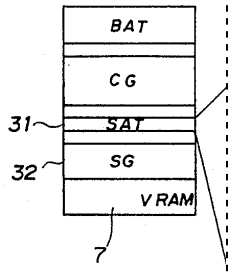
[Figure 4B] is an explanatory
apparatus of [Figure 1].

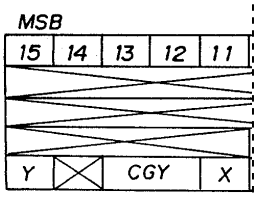
In [Figures 5A and 5B], there are shown background attribute tables (BATs) in the *VRAM* (7) each of 16 bits to have a character code of lower 12 bits for designating a pattern number of a character and a CG color of upper 4 bits for designating a CG color code.



[Figure 5A and 5B] are explan
VRAM in the apparatus of [Fig

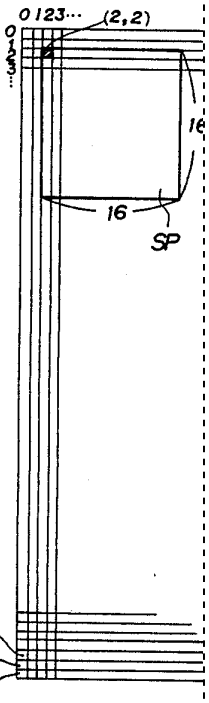
[Figures 6A and 6B]





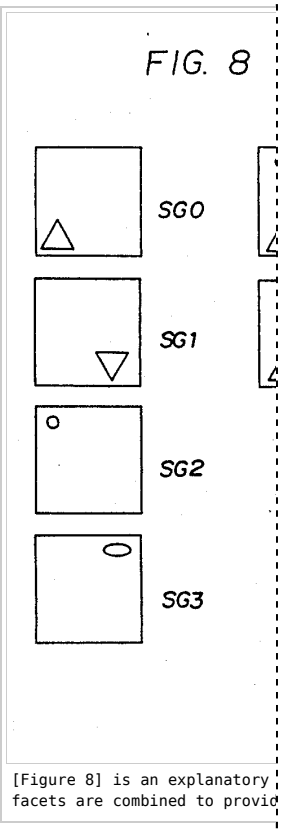
[Figures 6A and 6B] are expla
VRAM in the apparatus of [Fig

[Figure 7]



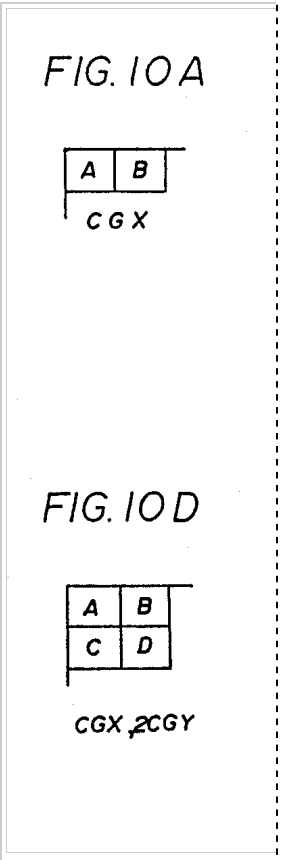
[Figure 7] is an explanatory
on a screen in the apparatus

[Figure 8]



[Figure 8] is an explanatory
facets are combined to provic

[Figure 10]



[Figure 10A to 10E] are expla
sprite is enlarged in the app

In [Figure 11A] , there is shown an apparatus for the control of an access to a video memory in an embodiment according to the invention. The apparatus for the control of an access to a video memory comprises an *oscillator* (51) for producing oscillation signals, a *frequency divider* (52) for dividing a frequency of the oscillation signals by a predetermined dividing ratio to produce pixel clock signals, a *memory width register* (3K) as already explained in [Figure 3K] having a content of a number of pixel periods dependent on a memory speed of the *VRAM* (7), a number of *pixel periods decision circuit* (53) for deciding a pixel width in accordance with the content of the *memory width register* (3K), means (54, 55 and 56) for producing a CPU address signal, DMA address signal and CG address signal respectively to designate addresses in an

access to the *VRAM* (7), an *address selector* (57) for selecting an address at an access timing which is set by the number of *pixel periods decision circuit* (53), and a data *latch circuit* (58) for latching data read from the *VRAM* (7). The *VRAM* (7) is shown in [Figure 11A] to include a *VRAM region* (33) of a fictitious screen as described in [Figure 4A] and a *Character Generator* (34) which is also shown in [Figure 11B]. One character of the *Character Generator* (34) is composed of four facets CH0 Ch1, CH2 and CH3 each having 8×8 pixels by which a pattern is defined by 16 words of 8 words for the facets CH0 and CH1 and other 8 words for the facets CH2 and CH3. The characters are addressed by the first addresses of the facets CHOs shown by A0, A1, A2 . . . which are defined by character codes of background attribute tables as described in [Figures 5A and 5B].

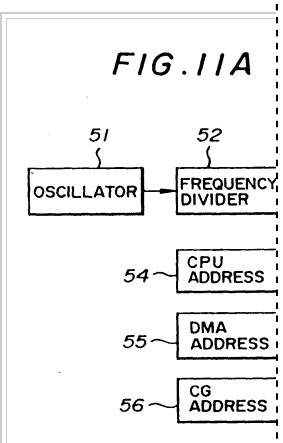
In operation, when a horizontal display of the scanning raster number *0* is started, the VM bits of the *memory width register* (3K) are checked by the number of *pixel periods decision circuit* (53). If it is assumed that a content of the VM bits is *00*, a number of pixel periods of an access to the *VRAM* (7) is decided to be *1* as defined in the table on page 19. Accordingly, the *VRAM* (7) is accessed in accordance with a CPU address signal from the *CPU address signal means* (54) under the control of the *address selector* (57) at the first pixel timing among 8 pixels of one character cycle. Next, the *VRAM region* (33) of the *VRAM* (7) is accessed at an address *0* in accordance with a CG address signal from the *CG address signal means* (56) at the second pixel timing. At this moment, a character code and a CG color are read from a background attribute table, as shown in [Figures 5A and 5B], of the address *0*. Thereafter, accesses are performed from the *CPU* (2 in [Figure 1]) to the *VRAM* (7) at the third and fifth pixel timings except for the fourth pixel timing, and the *Character Generator* (34) is accessed at the sixth pixel timing. In the access to the *Character Generator* (34), a CG address signal of the *CG address signal means* (56) is determined in accordance with a pattern number corresponding to a character code which is previously checked whereby display data are read from the facets CH0 and CH1 thereof. After the seventh pixel timing, display data are further read from the facets CH2 and CH3 in accordance with the same address signal at the 8 pixel timing. As a result, one character is formed in accordance with the display data of the four facets CH0 to CH3 which are latched in the data *latch circuit* (58). A display of the address *0* is performed on the *video display* (9 in [Figure 1]) in accordance with the data thus latched in the data *latch circuit* (58) wherein a color of the display is determined by the CG color in the background attribute table. In the horizontal displays which follow the horizontal display of the scanning raster line *0*, the same operation as explained above will be repeated.

On the other hand, if it is assumed that a content of the VM bits is *01*, *10* or *11*, the *VRAM* (7) is accessed with a number of pixel periods is *2*, *2* or *4*. For this reason, a content of the VM bits is determined dependent on a memory speed of the *VRAM* (7).

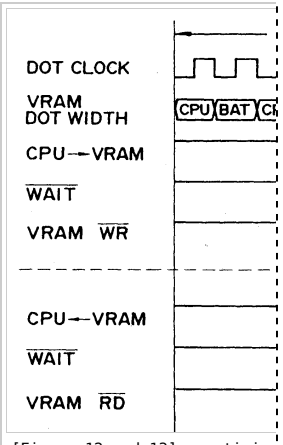
In another operation, it is assumed that a content of the VM bits is *00* in the *memory width register* (3K) to provide a number of pixel periods of one pixel, and that a content of the IW bits is *00* in the control register as shown in [Figure 3G] to provide an address increment width *1*. An operation in which data are written into the *VRAM* (7) is started after the first address for writing the data is written into the memory address write register as shown in [Figure 3C]. When the *VRAM* (7) is accessed from the *CPU* (2) as shown in [Figure 12] by the indication *CPU→VRAM*, the data are held in the *CPU read/write buffer* (22 in [Figure 2A]), if the writing of the data is collided in regard to its timing with a display cycle of the *VRAM* (7). For this reason, the *CPU* (2) is released from the writing of the data as changed from *0* to *1* in regard to a timing chart of *CPU→VRAM*. Thereafter, when the display cycle of the *VRAM* (7) is finished, the data thus held in the *CPU read/write buffer* (22) are written, as shown by the indication *VRAM* $\overline{WR}$, into the *VRAM* (7) at the address which is designated by the memory address write register of [Figure 3C]. At this moment, a content of the memory address write register is incremented by one. On the other hand, in a case where the *VRAM* (7) is accessed from the *CPU* (2) when data are held in the *CPU read/write buffer* (22), the condition $\overline{WAIT}$ becomes effective in the *CPU* (2) so that a wait signal is produced at the $\overline{BUSY}$ terminal connected to the *control unit* (20 in [Figure 2A]). Therefore, the condition $\overline{WAIT}$ is much decreased in the number of occurrences in accordance with the provision of the *CPU read/write buffer* (22).

In the same manner as described above, the *VRAM* (7) is accessed from the *CPU* (2) so that data are read from the VRAM as shown in [Figure 12] by the indication *CPU ← VRAM*. Data which are read from the *VRAM* (7) at an address designated by the memory address read register as shown in [Figure 3D] are once held in the *CPU read/write buffer* (22), when the access to the *VRAM* (7) is collided with the display cycle of the *VRAM* (7). When the display cycle of the *VRAM* (7) is finished, the data thus held in the *CPU read/write buffer* (22) are transferred to the *CPU* (2) as shown by the indication *VRAM* $\overline{RD}$. At this moment, a content of the memory address read register is incremented by one.

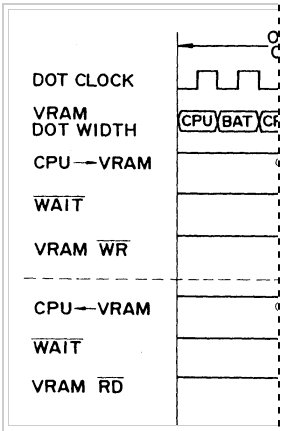
Otherwise, if the *CPU read/write buffer* (22) is not provided, the condition $\overline{WAIT}$ is increased in the number of occurrences as shown in [Figure 13] so that a throughput of the *CPU* (2) is decreased. In more detail, the condition $\overline{WAIT}$ is continued in the *CPU* (2) until the display cycle of the *VRAM* (7) is finished, when the *VRAM* (7) is accessed from the *CPU* (2) for the writing of data (*CPU→VRAM*) and the reading thereof (*CPU ← VRAM*) during that cycle. When the display cycle of the *VRAM* (7) is finished, data are written into the *VRAM* (7) as shown by the indication *VRAM* $\overline{WR}$, and read from the VRAM as shown by the indication *VRAM* $\overline{RD}$ whereby flickers are prevented from being occurred on a screen.



[Figure 11A] is a block diagr
video memory in an embodiment



[Figure 12 and 13] are timing
and a conventional apparatus
respectively.



[Figure 12 and 13] are timing and a conventional apparatus respectively.

Patent for Block Transfers Instructions

Full Patent

Original Patent (https://www.google.com/patents/US5034886).

United States Patent Number: **5034886**

COMPUTER SYSTEM FOR REDUCING NUMBER OF DEDICATED REGISTERS USING MEMORY STOCK AND SHARING OF ADDRESS AND GENERAL PURPOSE REGISTERS

File:US5034886-fig1.png

In [Figure 1], there is shown an apparatus for displaying a color image to which an apparatus for controlling a transfer of data according to the invention is applied. In the apparatus for displaying a color image, a *CPU* (1) performs a predetermined control in accordance with a program stored in *ROM* (5) so that data, arithmetical results etc. are stored into a *RAM* (6) temporarily. A *Video Display Controller* (2) is provided therein to supply a *Video Color Encoder* (3) with video data of a story, for instance, for a so-called television game read from a video RAM (VRAM) 7 in accordance with a control of the *CPU* (1) which deciphers a program for the television game stored in the *ROM* (5). The *Video Color Encoder* (3) to which the video data are supplied produces RGB analog signals obtained in accordance with color data stored therein, or produces video color signal including a luminance signal and color difference signals obtained in accordance with the color data. Further, a *Programmable Sound Generator* (4) is provided therein to produce analog sound signals as left and right stereo sounds in accordance with a content of the *ROM* (5) which is supplied through the *CPU* (1) thereto. The video color signal produced in the *Video Color Encoder* (3) is supplied through an interface 8 to a receiving circuit of a *video display* (9) as a composite signal, and the RGB analog signal is supplied through an *interface* (10) directly to a CRT of the television set which functions as an exclusive use monitor means. On the other hand, the left nd right analog sound signals are supplied through amplifiers 11a and 11b to speakers 12a and 12b to produce sounds.

[Figure 2] shows the *CPU* (1) and the *Programmable Sound Generator* (4) as encircled by a dotted line in [Figure 1]. The *CPU* (1) in which an apparatus for controlling a transfer of data in the embodiment is included and comprises:

- an *instruction register* (20)
- an *instruction decoder* (21)
- a *bus interface register* (22)
- an *Arithmetic Logic Unit* (ALU) (23)
- a *set of registers* (24)
- a *Mapping Register* (25)
- a *chip enable decoder* (26)
- a *timing and control unit* (27)
- an *input and output port* (28)
- a *Timer* (29)
- an *interrupt request register* (30)
- an *interrupt disable register* (31)
- and so on.

These units will be explained as follows. [list=1] [*] *instruction register* (20) The *instruction register* (20) is loaded with an instruction code at an instruction fetch cycle.

[*] *instruction decoder* (21) The *instruction decoder* (21) performs a sequential operation determined in accordance with an output of the *instruction register* (20), an interruption input from a peripheral circuit or a reset input, and further performs a control of a divergence command changing a flow of a program in accordance with informations of a status register described later.

[*] *bus interface register* (22) The *bus interface register* (22) controls a transfer of data among a *B-bus* (32), a *U-bus* (33) and an external bus D0 to D7. The *ALU* (23) and the *set of registers* (24) are connected by the *B-bus* (22), and the *U-bus* (33) is connected to internal

periphery circuits. Further, a *L-bus* (34) for transferring lower 8 bits of a logical address and a *H-bus* (35) for transferring upper 8 bits of the logical address are provided. A logical address low register 48 is connected to the *L-bus* (34), and a logical address high register 49 is connected to the *H-bus* (35).

[*] *ALU* (23) The *ALU* (23) is provided with an *A register* (36) and a *B register* (37), and performs all of arithmetic and logic operation. The A and B registers 36 and 37 are loaded with one or two data so that an arithmetic operation is performed in accordance with a control signal of the *instruction decoder* (21) to supply one of the *B-bus* (32), *L-bus* (34) and *H-bus* (35) with a result of the arithmetic operation.

[*] *set of registers* (24) The *set of registers* (24) comprises following 10 registers each being of 8 bits.

[list=1] [*] *Accumulator* (38)

The *Accumulator* (38) is a wide use register which plays the most important role in an arithmetic and logic operation to be conducted when a memory arithmetic flag T of a status register described later is *0*. Data thereof is supplied to an input of the *ALU* (23), and a result of the arithmetic is stored therein. The *Accumulator* (38) is also used for a transfer of data between memories and between a memory and a peripheral circuit, and for a count of a data block length when a block transfer of data is performed. A lower data of the length are stored therein after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6).

[*] *X and Y registers* (39 and 40)

The *X and Y registers* (39 and 40) are wide use registers which are mainly used for an index addressing. The *X register* (39) is used for a designation of an address on page *0* of a memory which is a destination of an arithmetic operation, and for a storage of lower data of a source address after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed. On the other hand, the *Y register* (40) stores lower data of a destination address after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed.

[*] *Program Counters* (41 and 42)

An up counter of 16 bits is composed of the *Program Counter* (41) of upper 8 bits and the *Program Counter* (42) of lower 8 bits. The up counter is automatically incremented in accordance with the conduct of a command to designate an address of a command or operand to be next conducted. Contents of the *Program Counters* (41 and 42) are evacuated into a stack region of the *RAM* (6) in a case where a command of subroutine is conducted, and an interrupt is produced, or after an interruption command of a software is conducted.

[*] *Stack Pointer* (43)

The *Stack Pointer* (43) designates lower 8 bits of the highest address on a stack region of the *RAM* (6), and is decremented after the pushing of data into the stack region and incremented before the pulling of the data from the stack region. For instance, 256 bytes of addresses *0x2100* to *0x21FF* are allocated to the stack region in a logical address.

[*] *source high register* (45), *destination high register* (46), and *length high register* (47)

These registers function in case of a command of a block transfer. The *source high register* (45) provides an upper byte of a source address to designate the source address together with a content of the *X register* (39). The *destination high register* (46) provides an upper byte of a destination address to designate the destination address together with a content of the *Y register* (40). The *length high register* (47) provides lower 8 bits for a down counter together with upper 8 bits which are a content of the *Accumulator* (38) so that a length of a block transfer is counted by a byte unit. [/list] [*] *Mapping Register* (25) The *Mapping Register* (25) is composed of 8 registers each being of 8 bits to convert a logical address of 16 bits to a physical address of 21 bits, and is selected by upper 3 bits of the *H-bus* (35).

[*] *chip enable decoder* (26) The *chip enable decoder* (26) provides chip enable outputs for following peripheral circuits by decoding upper 11 bits of a physical address.

[list=1] [*] a chip enable for the *RAM* (6)...$\overline{\text{CER}}$ [*] a chip enable for the *Video Display Controller* (2)...$\overline{\text{CE7}}$ [*] a chip enable for the *Video Color Encoder* (3)...$\overline{\text{CEK}}$ [*] a chip enable for the *Programmable Sound Generator* (4)...$\overline{\text{CEP}}$ [*] a chip enable for the *Timer* (29)...$\overline{\text{CET}}$ [*] a chip enable for the input and output port...CE10 [*] a chip enable for the *interrupt request register* (30) and the *interrupt disable register* (31)...$\overline{\text{CECG}}$ [/list]

[*] *timing and control unit* (27) The *timing and control unit* (27) is connected to the following terminals.

[list=1] [*] $\overline{\text{RD}}$ terminal A read timing signal is supplied through the $\overline{\text{RD}}$ terminal at a reading cycle.

[*] $\overline{\text{WR}}$ terminal A write timing signal is supplied through the $\overline{\text{WR}}$ terminal at a writing cycle.

[*] SYNC terminal A synchronous signal of *High* is supplied through the SYNC terminal at an instruction fetch cycle, that of *Low* is supplied therethrough at a system reset timing.

[*] $\overline{\text{NMI}}$ terminal A non-maskable interruption is produced when $\overline{\text{NMI}}$ input signal is supplied through the $\overline{\text{NMI}}$ terminal. A sub-routine call is conducted by reading lower address from the logical address *0xFFFC* and upper address from the logical address *0xFFFD* when a command which is conducted in a program is completed.

[*] $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ terminals

A sub-routine call is conducted by reading lower address from the logical address *0xFFF8* and upper address from the logical address *0xFFF9* when $\overline{\text{IRQ1}}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

A sub-routine call is conducted by reading lower address from the logical address *0xFFF6* and upper address from the logical address *0xFFF7* when $\overline{\text{IRQ2}}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in

the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

[*] $\overline{\text{RESET}}$ terminal A program is started by reading lower address from the physical address *0x00001FFE* and upper address from the physical address *0x00001FFF* when a $\overline{\text{RESET}}$ input becomes *Low*.

[*] RDY terminal The *CPU* (1) is started to operate when a RDY input is changed from *Low* to *High*.

[*] SX terminal A complementary signal of a system clock signal is supplied through the SX terminal.

[*] OSCI terminal An external clock signal is input through the OSCI terminal.

[*] $\overline{\text{EA1}}$ to $\overline{\text{EA3}}$ terminals These are input terminals for a test of the *CPU* (1).

[*] HSM terminal A speed mode signal of *High* is supplied through the HSM terminal in case of a high speed mode of 21.477270 MHz/3, and that of *Low* is supplied therethrough in case of a low speed mode of 21.477270 MHz/12. [/list] [*] *input and output port* (28) The *input and output port* (28) is connected to following terminals.

[list=1] [*] K0 to K7 terminals The terminals are input ports from which data are written in accordance with the conduct of a reading cycle in regard to the physical addresses *0x001FF000* to *0x001FF3FF*.

[*] 00 to 07 terminals The terminals are output ports with latches to which data are supplied in accordance with the conduct of a writing cycle in regard to the physical addresses *0x001FF000* to *0x001FF3FF*. [/list]

[*] *Timer* (29)

The *Timer* (29) is connected to a test input terminal EAT for the *CPU* (1) and provides a timer signal through the U-bus thereto.

[*] *interrupt request register* (30) The *interrupt request register* (30) is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *1* to make the $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ terminals *Low* and the remaining 1 bit is *1* to produce a timer interrupt signal. The *interrupt request register* (30) is only used for *read*.

[*] *interrupt disable register* (31) The *interrupt disable register* (31) is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *1* to make an interrupt request of the $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ terminals disable, and the remaining one is *1* to make an interrupt request disable in accordance with the timer interrupt signal. [/list]

In operation, when one of commands TII, TIN, TIA, TAI and TDD for a block transfer of data as shown in [Figure 3] is produced, contents of the *Accumulator* (38), the *X register* (39) and the *Y register* (40) are evacuated into a stack region of the *RAM* (6). Thereafter, the *Accumulator* (38) stores lower data of a length for the block transfer, the *X register* (39) stores lower data of a source address, and the *Y register* (40) stores lower data of a destination address. Simultaneously, the *source high register* (45) stores upper data of the source address, the *destination high register* (46) stores upper data of the destination address, and the *length high register* (47) stores upper data of the length for the block transfer. Thus, corresponding registers are loaded with the source address, the destination address, and the block length respectively. At the present stage, the memory arithmetic flag T of the *Status Register* (44) is *0*. Next, the aforementioned block transfer commands TII, TIN, TIA, TAI and TDD will be explained in conjunction with [Figure 3] to 7.

[list=1] [*] TII ([Figures 3 and 4]) In accordance with the command *TII*, data are transferred in a block of a predetermined length such that the source and destination addresses are automatically incremented. At first, contents of the *Accumulator* (38), the *X register* (39), and *Y register* (40) are evacuated into the stack region M_S as shown in [Figure 4] (block 410) by M_S *â†* *Y*, M_S *â†* *A* and M_S *â†* *X*, and the *Stack Pointer* (43) is decremented after the pushing of data into the stack region M_S as shown in [Figure 4] (block 410) by *S â†* *S - 1*. Thus, the block transfer of data is performed from a memory M_{SS} designated by the source high and X registers (45 and 39) to a memory M_{DD} designated by the destination high and Y registers (46 and 40) as shown in [Figure 4] (block 420) by M_{DD} *â†* M_{SS}. During this transferring stage,the source and destination addresses are incremented by each transfer of one byte as shown in [Figure 4] (block 420) by *SL â†* *SL + 1*, *SH â†* *SH + C*, *DL â†* *DL + 1*, *DH â†* *DH + C*. When contents of the *length high register* (47) and the *Accumulator* (38) for the length counter becomes *0*, that is, *Low* is *0*, in accordance with a down count as shown in [Figure 4] (block 430, block 420) by *LL â†* *LL - 1*, *LH â†* *LH - C*, the block transfer is completed. At this moment, data which have been evacuated in the stack region M_S are restored in the *Accumulator* (38), the *X register* (39), and the *Y register* (40) as shown in [Figure 4] (block 440) by *X â†* M_S, *A â†* M_S and *Y â†* M_S, and the *Stack Pointer* (S) is incremented as shown in [Figure 4] (block 440) by *S â†* *S + 1*.

[*] TIN ([Figures 3 and 5]) Although like operating steps are indicated by like expressions between [Figures 4 and 5], the difference is that an address of a source memory is incremented by each transfer of one byte, while an address of a destination address is fixed (block 510).

[*] TIA ([Figures 3 and 6]) Although an address of a source memory is incremented by each transfer of one byte, an address of a destination memory is incremented and decremented alternately by each transfer of one byte (blocks 610 and 620).

[*] TAI ([Figure 3]) Although a flow chart is not shown for the embodiment, an address of a source memory is incremented and decremented alternately by each transfer of one byte, an address of a destination memory is only incremented.

[*] TDD ([Figures 3 and 7]) Both addresses of a source memory and a destination memory are decremented by each transfer of one byte (block 710). [/list] In the embodiments in which increment and decrement of an address are alternately performed, it becomes easy to set up an interface between the apparatus of the invention and a peripheral integrated circuit.

Patent for Store Immediate Instructions

Full Patent

Original Patent (https://www.google.com/patents/US5226140)

United States Patent Number: **5226140**

APPARATUS FOR CONTROLLING THE TRANSFER OF DATA

File:US5226140-fig1.png

In [Figure 1], there is shown an apparatus for displaying a color image to which an apparatus for controlling the transfer of data according to the invention is applied. In the apparatus for controlling the transfer of data, a *CPU* (1) performs a predetermined control in accordance with a program stored in *ROM* (5) so that data, arithmetical results etc. are stored into a *RAM* (6) temporarily. A *Video Display Controller* (2) is provided therein to supply a *Video Color Encoder* (3) with video data of a story, for instance, for a so-called television game read from a video RAM (VRAM) 7 in accordance with a control of the *CPU* (1) which deciphers a program for the television game stored in the *ROM* (5). The *Video Color Encoder* (3) to which the video data are supplied produces RGB analog signals obtained in accordance with color data stored therein, or produces video color signal including a luminance signal and color difference signals obtained in accordance with the color data. Further, a programmable sound generator 4 is provided therein to produce analog sound signals as left and right stereo sounds in accordance with a content of the *ROM* (5) which is supplied through the *CPU* (1) thereto. The video color signal produced in the *Video Color Encoder* (3) is supplied through an interface 8 to a receiving circuit of a *video display* (9) as a composite signal, and the RGB analog signal is supplied through an interface 10 directly to a *video display* (9) which functions as an exclusive use monitor means. On the other hand, the left and right analog sound signals are supplied through amplifiers 11a and 11b to speakers 12a and 12b to produce sounds.

File:US5226140-fig2.png

[Figure 2] shows the *CPU* (1) and the programmable sound generator 4 as encircled by a dotted line in [Figure 1]. The *CPU* (1) in which an apparatus for controlling a transfer of data in the embodiment is included and comprises an *instruction register* (20), an *instruction decoder* (21), a *bus interface register* (22), an *Arithmetic Logic Unit* (ALU) (23), a *set of registers* (24), a *Mapping Register* (25), a *chip enable decoder* (26), a *timing and control unit* (27), an *input and output port* (28), a *Timer* (29), an *interrupt request register* (30), an *interrupt disable register* (31), and so on. These units will be explained as follows.

1. *Instruction Register* (20) The *Instruction Register* (20) is loaded with an instruction code at an instruction fetch cycle.

2. *Instruction Decoder* (21) The decoder 21 performs a sequential operation determined in accordance with an output of the *instruction register* (20), an interrupt input from a peripheral circuit or a reset input, and further performs a control of a branch command changing a flow of a program in accordance with informations of a status register described later.

3. *Bus Interface Register* (22) The *Bus Interface Register* (22) controls a transfer of data among a *B-bus* (32), a *U-bus* (33) and an external bus D0 to D7. The *ALU* (23) and the *set of registers* (24) are connected by the *B-bus* (22) and the *U-bus* (33), and is connected to internal peripheral circuits. Further, a *L-bus* (34) for transferring lower 8 bits of a logical address and a *H-bus* (35) for transferring upper 8 bits of the logical address are provided. A *logical address low register* (48) is connected to the *L-bus* (34), and a *logical address high register* (49) is connected to the *H-bus* (35).

4. *ALU* (23) The *ALU* (23) is provided with an *A register* (36) and a *B register* (37), and performs all of arithmetic and logic operation. The A and B registers 36 and 37 are loaded with one or two data so that an arithmetic operation is performed in accordance with a control signal of the *instruction decoder* (21) to supply one of the B, L and H-buses 32, 34 and 35 with a result of the arithmetic operation.

5. *Set of Registers* (24) The *set of registers* (24) comprises the following 10 registers each being of 8 bits.
   a. *Accumulator* (38)

      The *Accumulator* (38) is a wide use register which plays the most important role in an arithmetic and logic operation to be conducted when a memory arithmetic flag T of a status register described later is *0*. Data thereof is supplied to an input of the *ALU* (23), and a result of the arithmetic is stored therein. The *Accumulator* (38) is also used for a transfer of data between memories and between a memory and a peripheral circuit, and for a count of a data block length when a block transfer of data is performed. A lower data of the length are stored therein after data stored therein are evacuated into a stack region of the *RAM* (6).

   b. *X and Y Registers* (39 and 40)

      The *X and Y Registers* (39 and 40) are wide use registers which are mainly used for an index addressing. The *X register* (39) is used for a designation of an address on page *0* of a memory which is a destination of an arithmetic operation, and for a storage of lower data of a source address after data stored therein are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed. On the other hand, the *Y register* (40) stores lower data of a destination address after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed.

   c. *Program Counters* (41 and 42)

      An up counter of 16 bits is composed of the *Program Counter* (41) of upper 8 bits and the *Program Counter* (42) of lower 8 bits. The up counter is automatically incremented in accordance with the conduct of a command to designate an address

of a command or operand to be next conducted. Contents of the counters 41 and 42 are evacuated into a stack region of the *RAM* (6) in a case where a command of subroutine is conducted, and an interrupt is produced, or after an interruption command of a software is conducted.

d. *Stack Pointer* (43)

The *Stack Pointer* (43) designates lower 8 bits of the highest address on a stack region of the *RAM* (6), and is decremented after the pushing of data into the stack region and incremented before the pulling of the data from the stack region. For instance, 256 bytes of addresses *0x2100* to *0x21FF* are allocated to the stack region in a logical address.

e. *Source High Register* (45), *Destination High Register* (46), and *Length High Register* (47)

These registers function in case of a command of a block transfer. The *source high register* (45) provides an upper byte of a source address to designate the source address together with a content of the *X register* (39). The *destination high register* (46) provides an upper byte of a destination address to designate the destination address together with a content of the *Y register* (40). The *length high register* (47) provides upper 8 bits for a down counter together with a content of the *Accumulator* (38) so that a length of a block transfer is counted by a byte unit.

6. *Mapping Register* (25) The *Mapping Register* (25) is composed of 8 registers each being of 8 bits to convert a logical address of 16 bits to a physical address of 21 bits, and is selected by upper 3 bits of the H bus 35.

7. *Chip Enable Decoder* (26) The *chip enable decoder* (26) provides chip enable outputs for following peripheral circuits by decoding upper 11 bits of a physical address.
   a. a Chip Enable for the *RAM* (6) . . . $\overline{CER}$
   b. a Chip Enable for the *Video Display Controller* (2) . . . $\overline{CE7}$
   c. a Chip Enable for the *Video Color Encoder* (3) . . . $\overline{CEK}$
   d. a Chip Enable for the Programable Sound Generator 4 . . . $\overline{CEP}$
   e. a Chip Enable for the *Timer* (29) . . . $\overline{CET}$
   f. a Chip Enable for the Input and Output Port . . . $\overline{CEIO}$
   g. a Chip Enable for the *Interrupt Request Register* (30) and the *Interrupt Disable Register* (31) . . . $\overline{CECG}$
8. *Timing and Control Unit* (27) The unit 27 is connected to following terminals.
   a. $\overline{RD}$ Terminal A read timing signal is supplied through the $\overline{RD}$ terminal at a reading cycle.
   b. $\overline{WR}$ Terminal A write timing signal is supplied through the $\overline{WR}$ terminal at a writing cycle.
   c. SYNC terminal A synchronous signal of *High* is supplied through the SYNC terminal at an instruction fetch cycle, that of *Low* is supplied therethrough at a system reset timing.
   d. $\overline{NMI}$ Terminal A non-maskable interrupt is produced when $\overline{NMI}$ input signal is supplied through the $\overline{NMI}$ terminal. A sub-routine call is conducted by reading lower address from the logical address *0xFFFC* and upper address from the logical address *0xFFFD* when a command which is conducted in a program is completed.
   e. $\overline{IRQ1}$ and $\overline{IRQ2}$ Terminals A sub-routine call is conducted by reading lower address from the logical address *0xFFF8* and upper address from the logical address *0xFFF9* when $\overline{IRQ1}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

      A sub-routine call is conducted by reading lower address from the logical address *0xFFF6* and upper address from the logical address *0xFFF7* when $\overline{IRQ2}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

   f. $\overline{RESET}$ Terminal A program is started by reading lower address from the physical address *0x001FFE* and upper address from the physical address *0x001FFF* when a $\overline{RESET}$ input becomes *Low*.
   g. RDY Terminal The *CPU* (1) is started to operate when a RDY input is changed from *Low* to *High*.
   h. SX Terminal A complementary signal of a system clock signal is supplied through the SX terminal.
   i. OSC1 Terminal An external clock signal is input through the OSC1 terminal.
   j. $\overline{EA1}$ to $\overline{EA3}$ Terminals These are input terminals for a test of the *CPU* (1).
   k. HSM Terminal A speed signal of *High* is supplied through the HSM terminal in case of a high speed mode of *21.477270 MHz*/3, and that of *Low* is supplied therethrough in case of a low speed mode of *21.477270 MHz*/12.
9. *Input and Output Port* (28) The *input and output port* (28) is connected to following terminals.
   a. K0 to K7 Terminals The terminals are input ports from which data are written in accordance with the conduct of a reading cycle in regard to the physical addresses *0x1FF000* to *0x1FF3FF*.
   b. 00 to 07 Terminals The terminals are output ports with latches to which data are supplied in accordance with the conduct of a writing cycle in regard to the physical addresses *0x1FF000* to *0x1FF3FF*.
10. *Timer* (29) The *Timer* (29) is connected to a test input terminal EAT for the *CPU* (1) and provides a timer signal through the U-bus thereto.
11. *Interrupt Request Register* (30) The register 30 is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *1* to show the $\overline{IRQ1}$ to $\overline{IRQ2}$ terminals *Low* and the remaining 1 bit is *1* to show a timer interrupt caused. The register 30 is only used for *read*.

12. *Interrupt Disable Register* (31) The register 31 is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *0* to show an interrupt request of the $\overline{IRQ1}$ and $\overline{IRQ2}$ terminals disable, and the remaining one is *0* to show an interrupt request disable in accordance with the timer interrupt.

In [Figure 3], there is shown the aforementioned *Mapping Register* (25) comprising 8 registers MPR0 to MPR7 each being of 8 bits, and connected through the *B-bus* (32) to an *input output controller* (50) and through an *output selector* (51) to the output terminals A13 to A20. The *output selector* (51) selects one register from the 8 registers MPR0 to MPR7 of the *Mapping Register* (25) in accordance with upper 3 bits H5 to H7 of upper data of a logical address on the *H-bus* (35).

File:US5226140-fig3.png

[Figure 4] shows a relation between the upper 3 bits H5 to H7 and one register selected from the 8 registers MPR0 to MPR7. If it is assumed that the upper 3 bits H5 to H7 are *010*, the register MPR2 is selected from the 8 registers MPR0 to MPR7. Data are read from the *Mapping Register* (25) by a command *TMAi* where *i* is an integer selected from *0* to *7*. For instance, data are read from the register MPR2 to be transferred through the *B-bus* (32) to the *Accumulator* (38) in accordance with a command "TMA_2 ". On the other hand, data are written into the *Mapping Register* (25) by a command *TAMi* where *i* is an integer selected from *0* to *7*. For instance, data are transferred to be written into the register MPR0 from the *Accumulator* (38) by a command TAM_0. The commands *TMAi* and *TAMi* are composed of two byte respectively, and lower byte thereof includes a bit of *1* corresponding in a bit number to a register number which is selected from the 8 registers MPR0 to MPR7 and remaining 7 bits of *0*. When one of the 8 registers MPR0 to MPR7 is selected in accordance with upper 3 bits of the *H-bus* (35), a content of the selected register is supplied through the output terminals A13 to A20 to a following stage so that a physical address of 21 bit is obtained together with a content of the *logical address low register* (48) to be supplied through the output terminals A0 to A7 thereto, and lower 5 bits of the *logical address high register* (49) to be supplied through the output terminals A8 to A12 thereto. When the most significant bit A20 of a physical address A0 to A20 is *1*, a command by which data designated in accordance with the physical address A0 to A20 are immediate-transferred to the *Video Display Controller* (2) is conducted. The command includes ST0, ST1 and ST2 by which codes are produced on A0 and A1 of the address bus at a write cycle as shown in [Figure 6]. The command is set in the *instruction register* (20).

File:US5226140-fig4.png

In operation, lower data of a logical address are set in the *logical address low register* (48), and lower 5 bits of upper data of the logical address are set in the *logical address high register* (49). Here, it is assumed that address data are set in the 8 registers MPR0 to MPR7 of the *Mapping Register* (25) respectively. When upper 3 bits of upper data of the logical address, that is to say, upper 3 bits H5 to H7 of the *H-bus* (35) are *001*, the register MPR1 is selected so that a content *F8* of the register MPR1 is supplied through the output terminals A13 to A20 to the following stage. These output signals are combined with output signals of the output terminals A8 to A12 and A0 to A7 to produce a physical address A0 to A20. Then, the *chip enable decoder* (26) decodes upper 11 bits A10 to A20 of the physical address A0 to A20 to produce a chip enable signal $\overline{CER}$ of *0* by which a data memory is enabled.

When the immediate data transfer command is produced, the signal $\overline{CE7}$ is *0*, the MSB A20 is *1*, and the bits A0 and A1 are of a content as shown in [Figure 6] so that data are transferred from the *CPU* (1) to the *Video Display Controller* (2) as shown in detail in [Figure 7].

[Figure 7] shows the *Video Display Controller* (2) which comprises a *control unit* (70), an *address unit* (71), a *CPU read/write buffer* (72), a *Sprite Attribute Table buffer* (73), a sprite attribute shift register 74, a *background shift register* (75), a *data bus buffer* (76), a *synchronous circuit* (77), and a *priority circuit* (78).

File:US5226140-fig7.png

The *control unit* (70) is provided with a $\overline{BUSY}$ terminal through which *0* is supplied to the *CPU* (1) in a case where the *Video Display Controller* (2) is not in time for the writing / reading of data from the *CPU* (1) to the *VRAM* (7) so that the *CPU* (1) is held under the state, a $\overline{IRQ}$ terminal through which an interrupt request signal is supplied to an external circuit, a CK terminal through which a clock signals having a frequency for one pixel (one picture element) is received, a $\overline{RESET}$ terminal through which a reset signal for initialization is received, and a EX8/16 terminal through which a data bus change-over signal for changing over a data bus width is received.

The *address unit* (71) is connected to address terminals MA0 to MA15 through which an address signal of the VRAM is supplied thereto. An address region of the *VRAM* (7) is of a capacity, for instance, 65,536 words, provided that one word is of 16 bits. The *address unit* (71), the CPU read/write buffer 72, the *Sprite Attribute Table* (73), the *sprite shift register* (74) and the *background shift register* (75) are connected through data bus to data terminals MD0 to MD15 through which data of the *VRAM* (7) are transferred for the writing thereinto or the reading therefrom.

The *Sprite Attribute Table buffer* (73) is a memory for storing a display position (X, Y), color, pattern number and so on of a sprite (16 bits).

The *sprite shift register* (74) is supplied with a pattern number, sprite color etc. read from the *Sprite Attribute Table buffer* (73) to access the *VRAM* (7), and stores data for a pattern, color etc. of a sprite read from a sprite generator in the *VRAM* (7).

The *background shift register* (75) stores a pattern and CG color of a background. The pattern is read from a character generator in accordance with an address of the character generator in the *VRAM* (7) obtained from a character code which is read from an attribute table in the *VRAM* (7) together with the CG color.

The *data bus buffer* (76) is connected to terminals D0 to D15 through which data are supplied to and received from an external circuit. The *Video Display Controller* (2) can select one of 8 bit interface and 16 bit interface in compliance with a data width of a system including the *CPU* (1). When the 8 bit interface is selected, the terminals D0 to D7 are used among the terminals D0 to D15.

The *synchronous circuit* (77) is connected to a DISP terminal through which a signal for indicating a display period is supplied to an external circuit, a $\overline{VSYNC}$ terminal through which a signal for a vertical synchronization of a CRT is supplied to the *video display* (9) and an external vertical synchronous signal is received, and a $\overline{HSYNC}$ terminal through which a signal for a horizontal synchronization of the CRT is supplied to the *video display* (9)

and an external horizontal synchronous signal is received.

The *priority circuit* (78) is connected to terminals VD0 to VD7 through which video data are supplied to an external circuit, and a *SP/BG* (VD8) terminal through which a signal *1* is supplied to an external circuit when the video data are for a sprite and a signal *0* is supplied thereto when the video data are for a background.

The aforementioned *control unit* (70) is further provided with a $\overline{CS}$ terminal through which a signal *0* is received so that registers therein are accessed for the writing and reading of data from the *CPU* (1), $\overline{RD}$ and $\overline{WR}$ terminals through which read write-timing signals are received, and terminals A0 and A1 connected to the address bus of the *CPU* (1) as shown in [Figure 2].

The *Video Display Controller* (2) is further provided with a $\overline{MRD}$ terminal and a $\overline{MWR}$ terminal. When the $\overline{MRD}$ terminal is under a state of *0*, data are read from the *VRAM* (7) to the *CPU* (1). On the other hand, when the $\overline{MWR}$ terminal is under a state of *0*, data are written into the *VRAM* (7) from the *CPU* (1).

The *control unit* (70) includes a memory address write register and a VRAM data write register together with other registers.

In operation, when an immediate data transfer command (ST0, ST1 or ST2) is produced, data transferred from the *CPU* (1) are stored in the VRAM data write register. The aforementioned physical address A0 to A20 is automatically set in the memory address write register without receiving an address setting command. As a result, the data are written into the *VRAM* (7) in synchronization with a write signal of *0* applied to the $\overline{WR}$ terminal in accordance with a single transfer command which is called "an immediate data transfer command".

That is to say, a chip including the aforementioned memory address write register and VRAM data write register is enabled in accordance with a chip enable signal $\overline{CE7}$ which becomes *0* in the conduct of the immediate data transfer command of ST0, ST1 or ST2. As a result, bits A0 and A1 of a physical address A0 to A20 of 21 bits becomes a bit state dependent on the command (ST0, ST1 or ST2). Accordingly, the data transferred from the *CPU* (1) are automatically stored in the *VRAM* (7) without the necessity that an address is set in the memory address write register by a command.

[Figure 8] shows a relation between a logical address of the *CPU* (1) and a physical address of the *RAM* (6) wherein a block *F8* of a physical address region having 2M bytes is allocated in accordance with a content *F8* of the selected register MPR1 in a case where a logical address of 16 bits on the L and H-buses 34 and 35 corresponds to one address selected from addresses *0x2000* to *0x3FFF* of a logical address region having 64K bytes as indicated by hatching lines therein. Thus, a memory of 2M bytes in which an address signal of 21 bits is required for the access thereof can be accessed by an address signal of 16 bits. At this time, data at a corresponding address of a memory are immediate-transferred to a *Video Display Controller* (2) in a case where lower 2 bits A0 and A1 of the physical address are of a content as shown in [Figure 6].

File:US5226140-fig8.png

[Figure 9] shows a physical address region of the *RAM* (6) to which a chip enable signal $\overline{CER}$ is allocated. The chip enable signal $\overline{CER}$ is produced in accordance with a content of upper 11 bits A10 to A20 of a physical address A0 to A20 which is decoded in the *chip enable decoder* (26). For instance, the chip enable signal $\overline{CER}$ is allocated to a region of physical addresses *0x1F0000* to *0x1F7FFF* in which zero page and stack regions exist in a case where a content of the selected register MPR1 is *F8*. The zero page and stack regions are of a region to which contents of the *Accumulator* (38), the *X register* (39), and the *Y register* (40) are evacuated temporarily.

Although the invention has been described with respect to specific embodiment for complete and clear disclosure, the appended claims are not to thus limited but are to be construed as embodying all modification and alternative constructions that may occur to one skilled in the art which fairly fall within the basic teaching herein set forth.

Patent for the Programmable Sound Generator

Full Patent

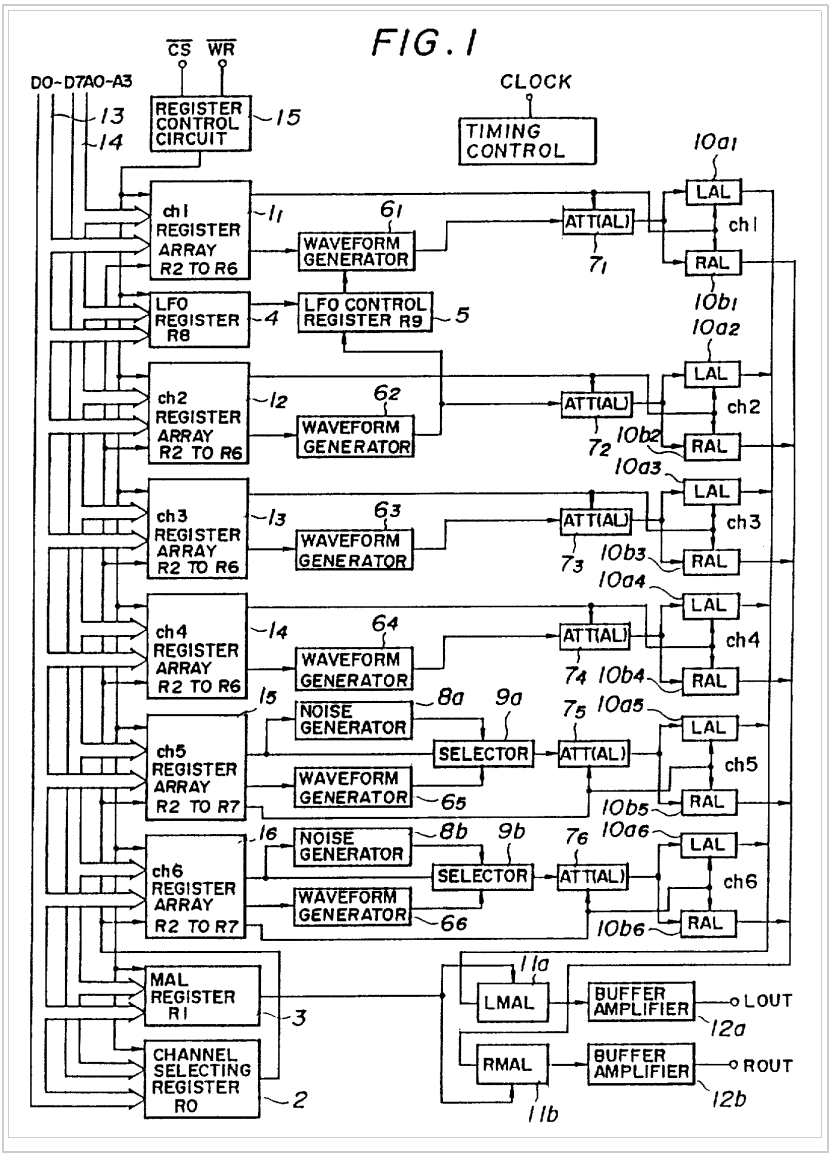
Original Patent (https://www.google.com/patents/US4924744)

United States Patent Number: **4924744**

APPARATUS FOR GENERATING SOUND THROUGH LOW FREQUENCY AND NOISE MODULATION

In [Figure 1], there is shown an apparatus for generating sound in an embodiment according to the invention in which six sound sources of channels 1 to 6 are provided. The channels 1 to 6 comprise register arrays 1_1 to 1_6, respectively, each including registers to be described later. The apparatus further comprises a *channel selecting register* (R0) (2), a *main sound volume adjusting register* (R1) (3), a *low frequency oscillator (LFO) frequency register* (R8) (4), and a *low frequency oscillator (LFO) control register* (R9) (5). Each of the register

arrays 1_1 to 1_6 includes a *fine frequency adjusting register* (R2), a *rough frequency adjusting register* (R3), a *channel ON/sound volume register* (R4), a *left and right sound volume register* (R5), and a *waveform register* (R6). Even more, the register arrays 1_5 and 1_6 of the channels 5 and 6 further includes a *noise enable/noise frequency register* (R7). In the channels 1_1 to 1_4, there are provided waveform generators 6_1 to 6_4 for supplying output sounds to attenuators 7_1 to 7_4 in which the output sounds are converted from digital signal to analog signal, and adjusted to have predetermined sound volumes. On the other hand, one of outputs of a waveform generator 6_5 and a noise generator 8a, and one of outputs of a waveform generator 6_6 and a noise generator 8b are selected to be supplied to attenuators 7_5 and 7_6 in the channels 5 and 6 by selectors 9a and 9b. Outputs of the attenuators 7_1 to 7_6 are divided to be supplied to left and right attenuators $10a_1$ and $10b_1$ to $10a_6$ and $10b_6$ in which the outputs are attenuated with predetermined attenuation factors to be mixed separately by left and right sounds. The left and right mixed signals are adjusted in main attenuators 11a and 11b to have predetermined sound volumes, and then passed through buffer amplifiers 12a and 12b to be supplied through output terminals LOUT and ROUT to a following stage. The register arrays 1_1 to 1_6, the *channel selecting register* (R0) (2), the *main sound volume adjusting register* (R1) (3), the *low frequency oscillator (LFO) frequency register* (R8) (4), and the low frequency control register (R9) (5) are connected through *data bus* (D0) to D7 (8 bits) 13 and *address bus* (A0) to A3 (4 bits) 14 to CPU (not shown) thereby receiving address signals and data therefrom. Each register described above is connected to a register *control circuit* (15) to which a chip selecting signal $\overline{CS}$ and a writing instruction signal $\overline{WR}$0 are applied from the CPU so that data transferred through the *data bus* (13) from the CPU are written at an address designated by an address signal on the *address bus* (14) into one of the registers.



In [Figure 2], there are shown the aforementioned registers R0 to R9 each being of 8 bits which will be explained as follows.

(1) *channel selecting register* (R0) Channel selecting data *ch SEL* are stored in lower 3 bits.

	REGISTERS	D7	D6	D5	D4	D3	D2	D1	D0
R0	CHANNEL SELECTION	/	/	/	/	/	chSEL		
R1	MAIN VOLUME ADJUSTMENT	L MAL				R MAL			
R2	FINE FREQUENCY ADJUSTMENT	FRQ LOW							
R3	ROUGH FREQUENCY ADJUSTMENT	/	/	/	/	/	FRQ HIGH		
R4	CHANNEL ON, CHANNEL VOLUME	ch ON	DDA	/	AL				
R5	L, R SOUND VOLUME	LAL				RAL			
R6	WAVEFORM, DDA	/	/	/	WAVE DATA				
R7	NOISE ENABLE, NOISE FREQUENCY	NE	/	/	NOISE FRQ				
R8	LFO FREQUENCY	LFO FRQ							
R9	LFO CONTROL	LFTRG	/	/	/	/	/	LFCTL	

[Figure 3A] shows a relation between channel selecting data 0x0 to 0x5 and on of the channels 1 to 6 which is selected. For instance, if the channel selecting data are *011* (equal to *3*), the channel 4 "ch 4" is selected.

R0	0	1	2	3	4	5	6	7
CHANNEL	ch1	ch2	ch3	ch4	ch5	ch6	//	//

On the other hand, the registers R2 to R7 of the register arrays 1_1 to 1_6 and the other registers R1, R8 and R9 are addressed by the address signal A0 to A3 on the *address bus* (14). A relation thereof is shown in [Figure 3B]. The address signal is one of the values 0x0 to 0x9 dependent on a content of A0 to A3. For instance, if the address signal is *2* (*0010*), the *fine frequency adjusting*

A0~A3(HEX)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
REGISTER	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	/	/	/	/	/	/

register (R2) is selected.

(2) *main sound volume adjusting register* (R1) The left and right attenuators 11a and 11b are controlled to adjust volumes of the left and right mixed sounds by the register (R1). Upper 4 bits *LMAL* are data for defining attenuation amount of the left attenuator 11a, while lower 4 bits *RMAL* are that for the right attenuator 11b. When the 4 bit data is *0xF*, sound volume is the maximum, and is decreased by approximately 3 dB in a case where a set value of the register (R1) is decreased by *1*.

(3) *fine frequency adjusting register* (R2) Fine frequency adjusting data "FRQ LOW" of 8 bits are set.

(4) *rough frequency adjusting register* (R3) Rough frequency adjusting data "FRQ HIGH" of lower 4 bits are set. Here, data of 12 bits including the data "FRQ LOW" of 8 bits as lower data and the data "FRQ HIGH" of 4 bits as upper data are obtained. If it is assumed that the 12 bit data are *F*, a frequency f_1 of a waveform output is defined in each channel as follows: ##EQU1##

In the above equation, *7.15909 MHz* is a master frequency which is supplied from the CPU.

(5) *channel ON/sound volume register* (R4) Sound output of each channel and the writing of the *waveform register* (R6) which will be described later are controlled in accordance with "ch ON" of the MSB, a direct D/A mode which will be described later is controlled in accordance with *DDA* of the second bit, and attenuation amounts of the attenuators 7_1 to 7_6 are controlled in accordance with *AL* of lower 5 bits

[Figure 4] shows the control which is conducted in accordance with a content of the MSB and second bit of the Register (R4). As apparent from the descriptions therein, sound output is supplied from a corresponding channel in a case where the MSB is *1*, while sound output is not supplied therefrom, and data on the *data bus* (13) are possible to be written into a corresponding *waveform register* (R6) in a case where the MSB "ch ON" is *0*. Further, an address counter for the *waveform register* (R6) is reset, and a direct D/A mode is possible to be performed in a case where the second bit *DDA* is *1*. The direct D/A mode is a mode in which data transferred through the *data bus* (13) from the CPU are passed through a corresponding one of the register arrays 1_1 to 1_6 and supplied directly to a corresponding one of the attenuators 7_1 to 7_6 without being passed through any of the waveform generators 61 to 66. In the corresponding attenuator, the data thus transferred are converted from digital signal to analog signal to be supplied through the output terminals LOUT and ROUT to a following stage.

chON	DDA	MODE	OPERATION
0	0	WRITING INTO WAVEFORM REGISTER	INCREMENT OF ADDRESS AT WRITING
0	1	WRITING INTO WAVEFORM REGISTER	RESET OF ADDRESS COUNTER FOR WAVEFORM REGISTER
1	0	MIXING (SOUND OUTPUT)	ADDRESSING OF WAVEFORM REGISTER BY FREQUENCY OF FREQUENCY REGISTER
1	1	DIRECT D/A	RESET OF ADDRESS COUNTER, TRANSFER OF DATA TO D/A CONVERTER AT WRITING

	7	6	5	4	3	2	1	0	INTERNAL ADDRESS
R6	//	//	//	WAVE DATA					0
	//	//	//						1
	//	//	//						2
	//	//	//						3
	⋮	⋮	⋮						⋮
	//	//	//						1D
	//	//	//						1E
	//	//	//						1F

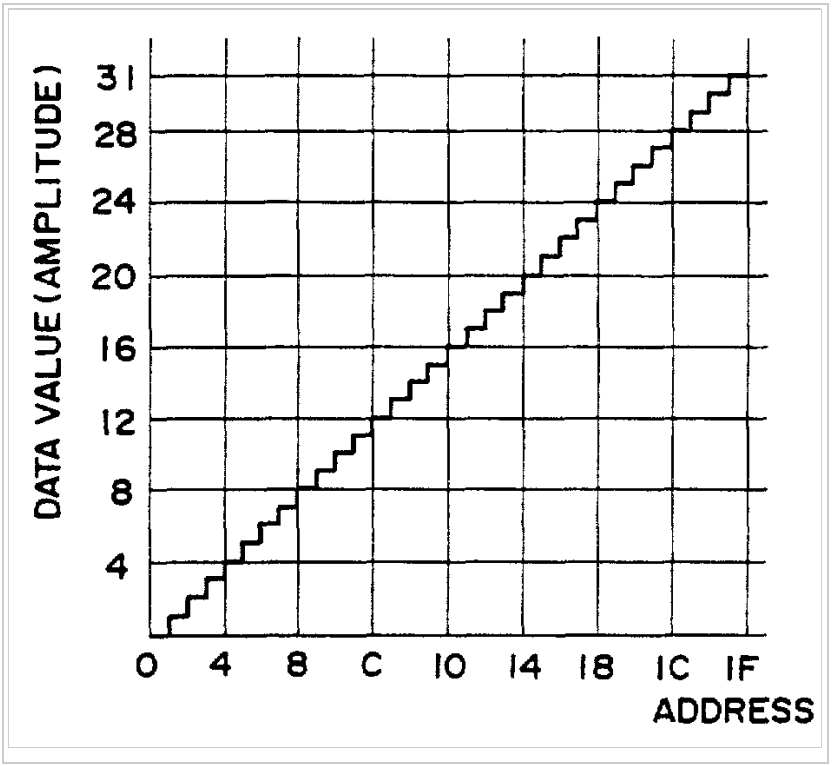
The lower 5 bits *AL* controls attenuation amounts of the attenuators 7_1 to 7_6 such that the maximum output is obtained when *0x1F* is set therein, and an amplitude of output is decreased by 1.5 dB each time when a set value is decreased by *1*.

(6) *left and right sound volume register* (R5) In each channel, the register (R5) controls a corresponding pair of the attenuators $10a_1$ and $10b_1$ to $10a_6$ and $10b_6$ for adjusting left and right dividing sound volumes. Sound volume of left output is decided by upper 4 bits *LAL*, while that of right output by lower 4 bits *RAL*. When the *LAL* or *RAL* is *0xF*, the sound volume is the maximum, and is decreased by approximately 3 dB each time when a set value is decreased by *1*.

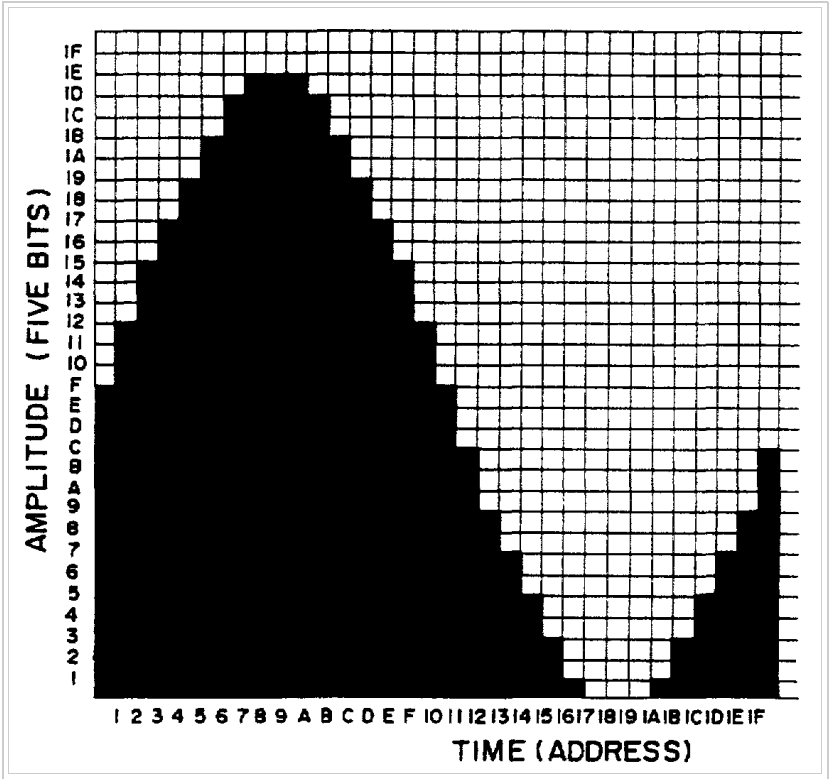
(7) *waveform register* (R6) The *waveform*

register (R6) is shown in [Figure 5] in which a word is composed of lower 5 bits for setting waveform data. As shown therein, the *waveform register* (R6) includes thirty-two addresses *0*, *1*, *2*, ... *1E*, *0x1F*, and thirty-two waveform data corresponding to the addresses and defining one period of a waveform. Therefore, the thirty-two addresses are accessed in an order of address numbers by a predetermined frequency and number so that a sound signal of a predetermined frequency is produced in a corresponding one of the waveform generators 61 to 66.

In [Figure 6], there is shown a relation between an address (horizontal axis) and an amplitude (vertical axis) in which the amplitude is increased from *0x00* to *0x1F* by one each time when the address is increased from *0x00* to *0x1F* by one. In a case where a waveform register having data as shown in [Figure 6] is accessed in an order of address numbers by a predetermined frequency, a sawtooth wave sound is generated.



[Figure 7] shows a data conversion in which a sinusoidal waveform of one period is divided equally by thirty-two addresses so that thirty-two amplitudes corresponding to the thirty-two addresses are expressed by 5 bit words which are stored into the *waveform register* (R6). Accordingly, when the *waveform register* (R6) is addressed in an order of address numbers, a waveform pattern as shown in [Figure 7] is generated in a corresponding one of the waveform generators 61 to 66. In a case where sound other than a sinusoidal waveform is generated, a waveform of the sound is observed in a waveform observing apparatus such as a synchroscope thereby being converted in regard to data thereof in the same manner as described in a case of a sinusoidal waveform.



The converted data are written into the *waveform register* (R6) in a following procedure. As explained in the *channel ON/sound volume register* (R4) in [Figure 4], a mode in which an address of the *waveform register* (R6) is increased by one each time when data are written thereinto in a case where both the upper 2 bits of the register (R4) are *0* ("ch ON" is *0*, and *DDA* is *0*). Thus, thirty-two words (thirty-two waveform data) are written into the thirty-two addresses thereof. At this stage, one of the channels 1 to 6 is selected by the *channel selecting register* (R0) (2), and *address data* (5) (A0 to A3) of the *address bus* (14) is controlled to be *6* so that data are transferred through the *data bus* (13) from CPU to the *waveform register* (R6). When data are finished to be written thereinto, the upper 2 bits of the register (R4) are controlled to be *10* ("ch ON" is *1*, and *DDA* is *0*) to provide an output mode. When the starting address of

the *waveform register* (R6) is wanted to be *0* at the moment that data are begun to be written thereinto, the upper 2 bits of the register (R4) are set to be *00* after the bits are once set to be *01*. Thus, an address counter for the *waveform register* (R6) is reset to be *0* in accordance with the control of software program.

In the aforementioned direct D/A mode, on the other hand, a preparation for generating sound is completed when the upper 2 bits of the register (R4) are set to be *00*, *00* is written into the *waveform register* (R6), and the upper 2 bits of the register (R4) are then set to be *11*. In the circumstance, when data are repeated to be written into the *waveform register* (R6), sound is generated because data are transferred to a corresponding one of the attenuators 7_1 to 7_6 (D/A converters) at each time of the data writing. In this case, although data are transferred to the *waveform register* (R6), the data may be considered to be passed therethrough. That is to say, a content of the *waveform register* (R6) remains unchanged to be maintained therein. By adopting this mode, a voice can be supplied through the apparatus from the CPU to a following stage in place of an artificial sound supplied from the *waveform register* (R6).

(8) *noise enable/noise frequency register* (R7) A change-over between noise sounds and musical sounds is controlled by the MSB bit *NE*. When the *NE* is *1*, the noise sounds are enabled to cease output of the musical sounds. Noise frequency is controlled by lower 5 bits. As described before, the noise generators 8a and 8b are provided in the channels 5 and 6. For this reason, the register (R7) is provided only in each of the channels 5 and 6. In the control of noise frequency, clock signal which is supplied to the noise generators 8a and 8b is controlled such that sound is shifted from low frequency to high frequency when a content of the lower 5 bits is varied from *0x00* to *0x1F*. The noise frequency f_2 is defined as follows:
##EQU2##

Where *NF* is a content of the lower 5 bits of the register (R7). The noise sound is a waveform of pseudo random, and output waveform is a rectangle wave which is applied to the generating of rhythm sound-effects sound.

(9) *low frequency oscillator (LFO) frequency register* (R8) (4)

A low frequency oscillator (LFO) is used for the control of frequency-modulation, and is composed of *low frequency oscillator (LFO) frequency register (R8)* (4) and a frequency counter for musical sounds in the channel 2. Here, it is defined that the frequency-modulation is a frequency-modulation in which sound of the channel 1 is frequency-modulated by using a waveform data of the channel 2. A frequency f_3 of the low frequency oscillator (LFO) is controlled by *low frequency oscillator (LFO) frequency register (R8)* (4), and the *fine frequency adjusting register* (R2) and the *rough frequency adjusting register* (R3) in the channel 2. Effects sound-vibrato is generated in accordance with the frequency modulation.

(10) *low frequency oscillator (LFO) control register* (R9) (5)

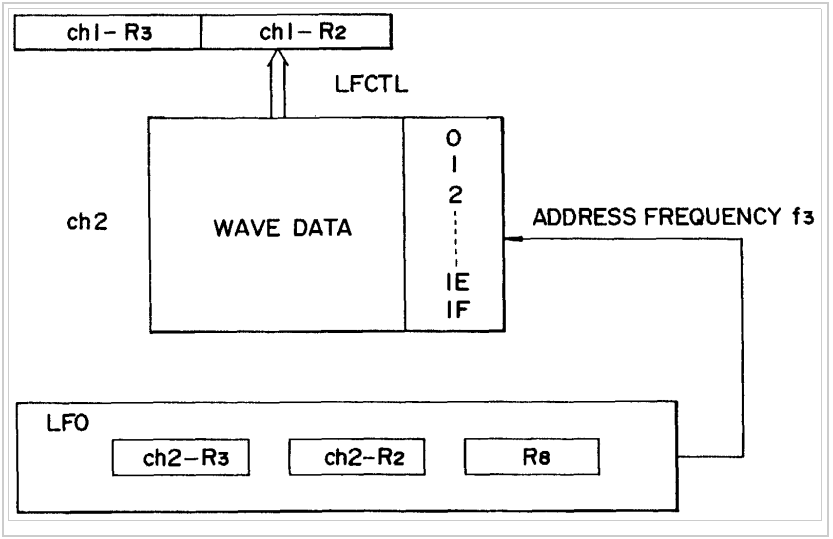
The *low frequency oscillator (LFO) control register* (R9) (5) includes the MSB "LF TRG" and lower 2 bits "LF CTL". A low frequency oscillator (LFO) is reset to return to the initial state when *1* is written into the "LF TRG". At this stage, a content at the address *0* of the *waveform register* (R6) in the channel 2 is an output which is used as frequency-modulation data, and the frequency-modulation is then stopped. While, the frequency-modulation is started when the "LF TRG" is *0*. That is to say, sound of

the channel 1 is frequency-modulated by waveform data of the *waveform register* (R6) in the channel 2 to generate effects soundvibrato.

The lower 2 bits "LF CTL" controls a modulation degree of the frequency-modulation.

[Figure 8] shows a relation between the low frequency oscillator (LFO) and the frequency-modulation. A frequency f_3 of the low frequency oscillator (LFO) which is an address frequency for addressing the *waveform register* (R6) in the channel 2 is defined as follows: ##EQU3##

Where F' is data value (decimal) of 12 bits including 8 bit data of the register (R2) in the channel 2 as lower data and 4 bit data of the register (R3) as upper data, and F" is data value (decimal) of 8 bits of the low frequency oscillator (LFO) register (R8) (4). The frequency-modulation of the channel 1 is performed by data of the *waveform register* (R6) which is addressed in the channel 2 by the address frequency f_3. That is to say, data of the *waveform register* (R6) in the channel 2 are added to or subtracted from the *fine frequency adjusting register* (R2) in the channel 1.



[Figure 9] shows frequency-modulation data (LFO data) corresponding to waveform data of waveform *0x00* to *0x1F* of the *waveform register* (R6) in the channel 2 whereby addition is performed in a case where the MSB of the waveform data is *1* (the waveform from *0x10* to *0x1F*), while subtraction is performed in a case where the MSB of the waveform data is *0x00* (the waveform *0x00* to *0x0F*). Sound of the channel 1 is shifted in a direction of low frequency by the addition, and in a direction of high frequency by the subtraction. For instance, when the waveform data of an address which is addressed in the *waveform register* (R6) of the channel 2 are *11100*, lower 4 bits *1100* (0xC) of the waveform data *11100* are added to four corresponding bits of the *fine frequency adjusting register* (R2) of the channel 1.

WAVE-FORM Hex	WAVEFORM BIT DATA	LFO DATA	WAVE-FORM Hex	WAVEFORM BIT DATA	LFO DATA	WAVE-FORM Hex	WAVEFORM BIT DATA	LFO DATA	WAVE-FORM Hex	WAVEFORM BIT DATA	LFO DATA
1F	11111	+F	17	10111	+7	F	01111	-1	7	00111	-9
1E	11110	+E	16	10110	+6	E	01110	-2	6	00110	-A
1D	11101	+D	15	10101	+5	D	01101	-3	5	00101	-B
1C	11100	+C	14	10100	+4	C	01100	-4	4	00100	-C
1B	11011	+B	13	10011	+3	B	01011	-5	3	00011	-D
1A	11010	+A	12	10010	+2	A	01010	-6	2	00010	-E
19	11001	+9	11	10001	+1	9	01001	-7	1	00001	-F
18	11000	+8	10	10000	0	8	01000	-8	0	00000	-10

[Figures 10A to 10C show 4 bits of the *fine frequency adjusting register* (R2) in the channel 1 to which the aforementioned lower 4 bits of the waveform data are added. That is to say, 4 bits of positions decided by a content of the lower 2 bits "LF CTL" of the *low frequency oscillator (LFO) control register* (R9) (5) are selected in the *fine frequency adjusting register* (R2) as follows:

1. *LF CTL = 0*

   The low frequency oscillator (LFO) is turned off, and output of normal musical sound is obtained.

2. *LF CTL = 1*

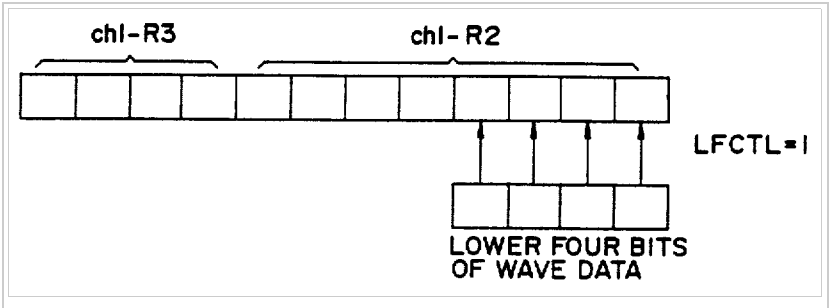
   Lower 4 bits of the *waveform register* (R6) of the channel 2 are added to or subtracted from lower 4 bits of the *fine frequency adjusting register* (R2) of the channel 1 to provide data by which a frequency of sound of the channel 1 is decided a shown in [Figure 10A].

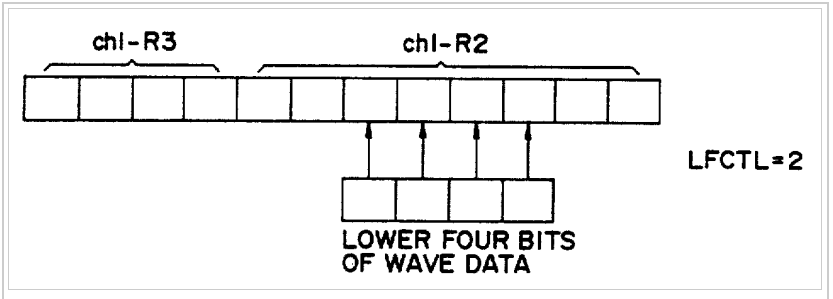
3. *LF CTL = 2*

   The lower 4 bits of the register (R6) of the channel 2 are added to or subtracted from middle 4 bits of the register (R2) of the channel 1 as shown in [Figure 10B].

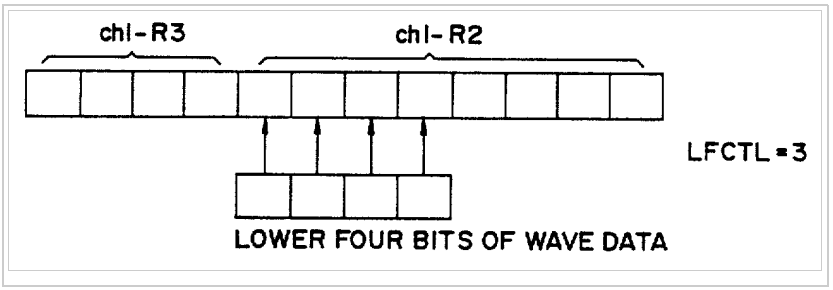
4. *LF CTL = 3*

   The lower 4 bit of the register (R6)

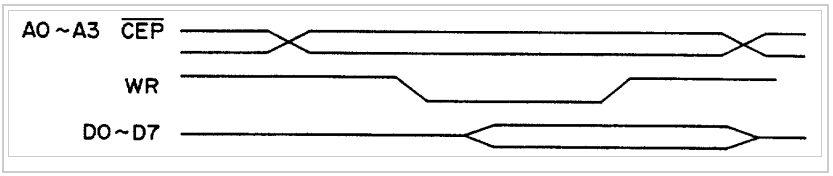




of the channel 2 are added to or subtracted from upper 4 bits of the register (R2) of the channel 1 as shown in [Figure 10C].



[Figure 11] shows a timing at which data are set in the registers (R0 to R9). Data are set in one of the registers R0, R1, R8 and R9 wherein a chip is enabled in accordance with *0* of $\overline{CEP0}$ signal, one of the registers is selected in accordance with the address A0 to A3 of the *address bus* (14), and data D0 to D7 which are set on the *data bus* (13) are written thereinto in accordance with *0* of the writing signal $\overline{WR}$. On the other hand, data are set in one of the registers R2 to R7 wherein one of the channels 1 to 6 is set in the *channel selecting register* (R0), and the same procedure as described above is thereafter performed to write data into a register in a channel thus selected.



Next, operations will be explained in the apparatus for generating sound in the embodiment according to the invention.

[OPERATION 1] It is assumed that predetermined data are already written in the registers (R0 to R9). In the circumstance, the whole system in the apparatus is enabled when the chip selecting signal $\overline{CS0}$ (=*0*) is applied to the register *control circuit* (15). When at least one of the channels and those of the registers are selected in accordance with a content of the *channel selecting register* (R0) and an address A0 to A3 on the *address bus* (14), the addresses *0x00* to *0x1F* of the *waveform register* (R6) are addressed in accordance with a frequency dependent on contents of the *fine frequency adjusting register* (R2) and the *rough frequency adjusting register* (R3) in the channel where the upper 2 bits of the *channel ON/sound volume register* (R4) are *10* so that output sound is generated in at least one of the waveform generators 61 to 66 in which waveform data of the *waveform register* (R6) are developed. The attenuation amounts of the attenuators 7_1 to 7_6 are set to be predetermined values in accordance with the lower 4 bits of the *channel ON/sound volume register* (R4), and the attenuation amounts of the attenuators 10a1 and 10b1 to 10a and 10b6 are set by the left and right (LR) sound volume register (R5). Output sound of each channel is converted from digital signal to analog signal in a corresponding one of the attenuators 7_1 to 7_6, and are adjusted to be a predetermined sound volume therein. The output sound is divided in each channel to be supplied to a corresponding pair of the attenuators $10a_1$ and $10b_1$ to $10a_6$ and $10b_6$, and the output sounds thus divided are mixed to provide left and right output sounds which are then controlled in the left and right main attenuators 11a and 11b set by the *main sound volume adjusting register* (R1) thereby providing the left and right output sounds with predetermined main sound volumes. The left and right stereo sounds of the main volumes thus adjusted are passed through the buffer amplifiers 12a and 12b to be supplied through the output terminals LOUT and ROUT to a following stage.

In the operation described above, when the MSB of the noise enable noise/frequency register (R7) is *1* in the channels 5 and 6, noise is generated in the noise generator 8a or 8b in accordance with a noise frequency of the lower 5 bits of the register (R7), and then selected to be

supplied to the attenuator 75 or 76 by the selectors 9a or 9b thereby producing rhythm sound-effects sound.

Further, the MSB of the *low frequency oscillator (LFO) control register* (R9) is changed from *1* to *0*, frequency-modulation of output sound is started in the channel 1 again. At this moment, an address frequency f_3 by which the *waveform register* (R6) is addressed is decided by a content of the low frequency oscillator (R9) and contents of the fine and rough frequency adjusting registers (R2 and R3). In accordance with the access of the *waveform register* (R6), lower 4 bits of the *waveform register* (R6) are added to or subtracted from selected 4 bits of the *fine frequency adjusting register* (R2) in the channel 1 dependent on the "LF CTL" (lower 2 bits) of the *low frequency oscillator (LFO) control register* (R9) to result in frequency-modulation of output sound thereby producing effects sound-vibrato.

On the other hand, when upper 2 bits of the *channel ON/sound volume register* (R4) are *11*, there is realized a direct D/A mode in which voice sound is supplied from the CPU through the *data bus* (13) to at least a corresponding one of the register arrays 1_1 to 1_6 each time when the writing signal $\overline{WR}0$ is applied to the register *control circuit* (15) in accordance with the resetting of the address counter for the *waveform register* (R6) to be transferred through a line which does not pass on a corresponding one of the *waveform generators* 6_1 to 6_6 to at least a corresponding one of the attenuators 7_1 to 7_6 in which voice sound is converted from digital signal to analog signal, and adjusted to be a predetermined sound volume thereby being supplied through the output terminals LOUT and ROUT in the same manner as described above. This results in the generation of voice sound which is utilized for effects sounds in place of artificial sound.

As apparent from the explanations, simultaneous generation of six musical sounds and of four musical sounds and two noise sounds and so on can be performed, although the number of waveform generators and noise generators is not limited to that of the embodiment.

[OPERATION 2] An address A0 to A3 of the *address bus* (14) are controlled to be *0000*, and the *channel selecting register* (R0) is addressed from the CPU as shown in [Figure 3B]. In the circumstance, *0* is written into lower 3 bits "ch SEL" thereof from the *data bus* (13) so that the channel 1b is selected as shown in [Figure 3A]. Next, the address A0 to A3 are controlled to be *0100*, and the *channel ON/sound volume register* (R4) is addressed in the channel 1. Under the situation, predetermined data are written into upper 2 bits "ch ON" and *DDA* of the register (R4) to conduct following operations.

(a) "ch ON"=*0*, and *DDA*=0 The *waveform register* (R6) is addressed in accordance with an address *0111* on the *address bus* (14). The addresses of the *waveform register* (R6) are serially addressed, when a counted value of the internal address counter is increased by one as shown in [Figure 5], so that waveform data transferred through the *data bus* (13) ar written thereinto. Such waveform data can be stored in the *waveform register* (R6) by performing a data conversion of a predetermined waveform as explained before. In this case, the writing of waveform data is started from a non-fixed address dependent on a then-counted value of the internal address counter.

(b) "ch ON"=*0*, and *DDA*=1 After upper 2 bits of the register R4 are set as *0* and *1*, the 2 bits are set to be *0* and *0* so that the address counter for the register R4 is reset to be *0*. Thereafter, data are

written thereinto in an order of the addresses *0x00* to *0x1F*. Thus, waveform data can be written into the *waveform register* (R6) in the channel 1. After the upper 2 bits of the *channel ON/sound volume register* (R4) are set to be *1* and *0*, respectively, so that output sound of the channel 1 is converted from analog signal to digital signal in the attenuator 71 and adjusted to have a predetermined sound volume, and then divided to be supplied to the attenuators 10a1 and 10b1 from which left and right output sounds of the channel 1 are supplied to the main attenuators 11a and 11b.

On the other hand, *0* (="ch ON") and *1* (=*DDA*) are written through the *data bus* (13) into the upper 2 bits of the *channel ON/sound volume register* (R4), and *00* are written into the *waveform register* (R6) so that the direct D/A mode is demanded. Thereafter, *1* (="ch ON") and *1* (=*DDA*) are written into the upper 2 bits of the register (R4) so that voice sound signals which are transferred through the *data bus* (13) are repeated to be written into the *waveform register* (R6), and supplied to the attenuator (D/A converter) 71 through a path which does not pass the waveform generator (R6) to provide voice sound output at each time of the writing of the *waveform register* (R6). In this case, although data are seemingly transferred to the *waveform register* (R6), a whole content of the *waveform register* (R6) is held to remain unchanged.

Although the operation of the channel 1 is explained above, the same operation is possible to be performed in the other channels 2 to 6 or in a plurality thereof simultaneously.

[OPERATION 3] When lower 2 bits "LF CTL" of the low frequency oscillator control register (R9) are not zero, that is, the bits are one of *1*, *2* and *3*, a low frequency oscillator is turned on to produce a low frequency signal f3. The frequency f_3 is calculated in the aforementioned equation, and depends on contents of the fine and rough frequency register (R2 and R3) and on a content of the *low frequency oscillator (LFO) frequency register* (R8). When *1* is written into the MSB "LF TRG" of the *low frequency oscillator (LFO) control register* (R9) (5), the low frequency oscillator (LFO) is reset to return to the initial state. At this moment, frequency-modulation is stopped in a state that waveform data at the address *0* of the *waveform register* (R6) in the channel 2 is read out as frequency-modulation data. In the circumstance, *0* is written into the MSB "LF TRG" of the control register (R9), frequency-modulation is started again with the low frequency signal f3 by which the *waveform register* (R6) of the channel 2 is serially addressed from the address *0* to the address *1F*. If the low frequency f_3 is high, an addressing speed is of a high speed, while the addressing speed is low if the low frequency f_3 is low. Waveform data of the *waveform register* (R6) in the channel 2 are accessed in an order of the addresses *0x00* to *0x1F* so that lower 4 bits of the waveform data are added to or subtracted from selected 4 bits of the *fine frequency adjusting register* (R2). At this moment, when the upper bit of the 5 bit waveform data is *1*, addition is performed, and when the upper bit is *0*, subtraction is performed. Here, when lower 2 bits "LF OTL" of the *low frequency oscillator (LFO) control register* (R9) is *01*, addition or subtraction is performed as shown in [Figure 10A], and when the "LF CTL" is *10* and *11* respectively, addition or subtraction is correspondingly performed as shown in [Figures 10B and 10C]. In this case, modulation degree is larger in [Figure 10B] than in [Figure 10A], and larger in [Figure 10C] than in [Figure 10B]. In this manner, when waveform data of the *waveform register* (R6) in the channel 2 are added to a

content of the *fine frequency adjusting register* (R2) in the channel 1, output sound of the channel 1 is frequency-modulated by the result of the addition or subtraction and a content of the *rough frequency adjusting register* (R3) thereby producing output sound of effects sound-vibrato in the channel 1.

[OPERATION 4] A mixing mode is set when *10* are set into upper 2 bits "ch ON" and *DDA* of the *channel ON/sound volume register* (R4) in each of the channel 1 to 6 where the *waveform register* (R6) is addressed with a frequency decided by the frequency adjusting registers (R2 and R3) so that output sound is generated in each of the waveform generators 1_1 to 1_6 of the channels 1 to 6. The output sound is converted from digital signal to analog signal and adjusted to be a predetermined sound volume decided by lower 5 bits *AL* of the *channel ON/sound volume register* (R4) in each of the attenuators (71 to 76). The output sound of the predetermined sound volume is than divided to be supplied to each left and right pair of the attenuators $10a_1$ and $10b_1$ to $10a_6$ and $10b_6$ in which the output sound is adjusted to be predetermined levels in accordance with contents of upper 4 bits *LAL* and lower 4 bits " RAL" of the *left and right sound volume register* (R5). Therefore, left output sounds of the attenuators 10a1 to 10a6 and right output sounds of the attenuators 10b1 to 10b6 are separately mixed with each other to be supplied to the left and right main attenuators 11a and 11b in which the whole left and right sounds are adjusted in regard to sound volume to be supplied through the buffer amplifiers 12a and 12b to the output terminals LOUT and ROUT. As a result, the left and right sound outputs of predetermined sound volumes are obtained at the output terminals LOUT and ROUT. The attenuation amounts of the *left and right main attenuators* (11a and 11b) are decided by the upper 4 bits *LMAL* and the lower 4 bits *RMAL* of the main sound volume register (R1). Accordingly, when it is assumed that the maximum sound volume is *0xF* , far and near sounds are generated in accordance with contents of the *LMAL* and *RMAL*.

[OPERATION 5] Upper 2 bits "ch ON" and *DDA* of the *channel ON/sound volume register* (R4) are set as *10* in a corresponding channel so that a mixing mode is set to address the *waveform register* (R6) with a frequency decided by the fine and rough frequency adjusting registers (R2 and R3). As shown in [Figure 5], waveform data are stored at the addresses *0x00* to *0x1F* of the *waveform register* (R6) so that waveform data are developed in an order of addresses at each time when a counted value of the address counter for counting address frequency is increased by one thereby producing waveform in a corresponding one of the waveform generators 61 to 66. In the corresponding channel, output sound is generated by repeating the addressing of the *waveform register* (R6) in a period of a waveform thus produced. Waveform of each channel is converted from digital signal to analog signal and adjusted to be a predetermined sound volume in each of the attenuators 7_1 to 7_6, and divided to be supplied to each pair of the left and right attenuators $10a_1$ and $10b_1$ to $10a_6$ and $10b_6$ in which left and right sounds are separately adjusted to be predetermined sound volumes. The left sounds of the channels are mixed to be supplied to the left main attenuator 11a, and the right sounds of the channels are mixed to be supplied to the right main attenuator 11b. The left and right sounds thus mixed are adjusted therein to be predetermined sound volumes and supplied through the buffer amplifiers 12a and 12b to the output terminals LOUT and ROUT.

The sound volume adjustment of the respective attenuators is performed in

accordance with contents of the *main sound volume adjusting register* (R1), the *channel ON/sound volume register* (R4), and the *left and right sound volume register* (R5). As described before, output sound volume is changed by 3 dB when contents of the register (R1 and R5) are changed by one, and the former is changed by 1.5 dB when the latter is changed by one.

Here, it is assumed that contents of the registers are as follows:

(a) *main sound volume adjusting register* (R1)

LMAL is *0xC*

RMAL is *0x8*

(b) *channel ON/sound volume register* (R4)

upper 4 bits of *AL* are *0xE*

sound volume is controlled in accordance with lower 1 bit of *AL* by 1.5 dB.

(c) *left and right sound volume register* (R5)

LAL is *0xF*

RAL is *0x8*

In accordance with the above assumptions, an attenuation value of the left output value will be calculated as follows because the maximum sound value is *0xF*.

(0xF - 0xC) + (0xF - 0xE) + (0xF - 0xF) = 0x4 4 × 3 dB = 12 dB Accordingly, a level down of 12 dB is resulted.

On the other hand, an attenuation value of the right output value will be calculated as follows:

(F-8)+(F-E)+(F-8)=15 15×3 dB=45 dB Accordingly, a level down of 45 dB is resulted. Here, if it is assumed that a dynamic range is 45 dB in a circuit, no output sound is obtained at the right output terminal ROUT.

Further, the *main sound volume adjusting register* (R1) can be used for fade in and fade out of a whole sound and for a left to right shift of the whole sound, while the *channel ON/sound volume register* (R4) can be used for an output level adjustment of a channel sound, and the *left and right sound volume register* (R5) can be used for a left and right distribution of a channel sound and for an output level adjustment of the channel sound.

[OPERATION 6] The *channel selecting register* (R0) is addressed in accordance with an address signal *0* (A0 to A3) on the *address bus* (14) so that data for selecting one of the channels 1 to 6 are set into lower 3 bits thereof. If the data are *0*, the channel 1 is selected. Next, the *left and right sound volume register* (R5) is addressed by *address data* (5) (A0 to A3) so that sound volume adjusting data are set into upper and lower 4 bits *LAL* and *RAL* respectively. Thus, sound volume adjusting levels are set into the left and right attenuators $10a_1$ and $10b_1$ to $10a_6$ and $10b_6$ of the channel 1 to 6. When upper 2 bits "ch ON" and *DDA* of the *channel ON/sound volume register* (R4) are set to be *10* in each channel, a mixing mode is set to address the *waveform register* (R6) with an address frequency decided by the frequency registers (R2 and R3). In accordance with the addressing of the *waveform register* (R6), waveform data are developed in the waveform generators 61 to 66 to produce output sounds which are converted from digital signals to analog signals and adjusted to be predetermined sound volumes in the attenuators 7_1 to 7_6. The output sounds of the attenuators 7_1 to 7_6 are divided to be supplied to the left and right attenuators $10a_1$ and $10b_1$ to $10a_6$ to $10b_6$ in which predetermined attenuation amounts decided by the *left*

and right sound volume register (R5) are given to output sounds of the channels. Thereafter, output sounds are supplied through the output terminals LOUT and ROUT to a following stage in the same manner as described before, although repeated explanations are omitted here.

Patent for High/Low Speed Modes

Full Patent

Original Patent (https://www.google.com/patents/US5483659).

United States Patent Number: **5483659**

APPARATUS FOR CONTROLLING A SIGNAL PROCESSING SYSTEM TO OPERATE IN HIGH AND LOW SPEED MODES

In [Figure 1], there is shown an apparatus for displaying a color image to which an apparatus for controlling a signal processing system (CPU) (1) according to the invention is applied. In the apparatus for displaying a color image, a *CPU* (1) performs a predetermined control in accordance with a program stored in *ROM* (5) so that data, arithmetical results etc. are stored into a *RAM* (6) temporarily. A *Video Display Controller* (2) is provided therein to supply a *Video Color Encoder* (3) with video data of a story, for instance, for a so-called television game read from a video RAM (VRAM) 7 in accordance with a control of the *CPU* (1) which deciphers a program for the television game stored in the *ROM* (5). The *Video Color Encoder* (3) to which the video data are supplied produces RGB analog signals obtained in accordance with color data stored therein, or produces video color signal including a luminance signal and color difference signals obtained in accordance with the color data. Further, a programable sound generator 4 is provided therein to produce analog sound signals as left and right stereo sounds in accordance with a content of the *ROM* (5) which is supplied through the *CPU* (1) thereto. The video color signal produced in the *Video Color Encoder* (3) is supplied through an interface 8 to a receiving circuit of a *video display* (9) as a composite signal, and the RGB analog signal is supplied through an interface 10 directly to a *video display* (9) which functions as an exclusive use monitor means. On the other hand, the left and right analog sound signals are supplied through amplifiers 11a and 11b to speakers 12a and 12b to produce sounds.

File:US5483659-fig.png

[Figure 2] shows the *CPU* (1) and the programable sound generator 4 as encircled by a dotted line in [Figure 1]. The *CPU* (1) in which an apparatus for controlling a signal processing system in the embodiment is included and comprises an *instruction register* (20), an *instruction decoder* (21), a *bus interface register* (22), an *Arithmetic Logic Unit* (ALU) (23), a *set of registers* (24), a *Mapping Register* (25), a *chip enable decoder* (26), a *timing and control unit* (27), an *input and output port* (28), a *Timer* (29), an *interrupt request register* (30), an *interrupt disable register* (31), and so on. These units will be explained as follows.

File:US5483659-fig.png

(1) Instruction Register 20 The register 20 is loaded with an instruction code at an instruction fetch cycle.

(2) Instruction Decoder 21 The decoder 21 performs a sequential operation determined in accordance with an output of the *instruction register* (20), an interrupt input from a peripheral circuit or a reset input, and further performs a control of a brunch command changing a flow of a program in accordance with informations of a status register described later.

(3) Bus Interface Register 22 The register 22 controls a transfer of data among a *B-bus* (32), a *U-bus* (33) and an external bus D0 to D7. The *ALU* (23) and the *set of registers* (24) are connected by the *B-bus* (22) and the *U-bus* (33), and is connected to internal peripheral circuits. Further, a *L-bus* (34) for transferring lower 8 bits of a logical address and a *H-bus* (35) for transferring upper 8 bits of the logical address are provided. A *logical address low register* (48) is connected to the *L-bus* (34), and a *logical address high register* (49) is connected to the *H-bus* (35).

(4) *ALU* (23) The *ALU* (23) is provided with an *A register* (36) and a *B register* (37), and performs all of arithmetic and logical operation. The A and B registers 36 and 37 are loaded with one or two data so that an arithmetic operation is performed in accordance with a control signal of the *instruction decoder* (21) to supply one of the B, L and H-buses 32, 34 and 35 with a result of the arithmetic operation.

(5) Set of Registers 24 The *set of registers* (24) comprises following 10 registers each being of 8 bits.

(a) *Accumulator* (38)

The *Accumulator* (38) is a wide use register which plays the most important role in an arithmetic and logical operation to be conducted when a memory arithmetic flag T of a status register described later is *0*. Data thereof is supplied to an input of the *ALU* (23), and a result of the arithmetic is stored therein. The *Accumulator* (38) is also used for a transfer of data between memories and between a memory and a peripheral circuit, and for a count of a data block length when a block transfer of data is performed. A lower data of the length are stored therein after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6).

(b) X and Y registers 39 and 40

The registers 39 and 40 are wide use registers which are mainly used for an index addressing. The *X register* (39) is used for a designation of an address on page *0* of a memory which is a destination of an arithmetic operation, and for a storage of lower data of a source address after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed. On the other hand, the *Y register* (40) stores lower data of a destination address after data stored therein at the very moment are evacuated into a stack region of the *RAM* (6) when a block transfer of data is performed.

(c) program counters 41 and 42

An up counter of 16 bits is composed of the *Program Counter* (41) of upper 8 bits and the *Program Counter* (42) of lower 8 bits. The up counter is automatically incremented in accordance with the conduct of a command to designate an address of a command or operand to be next conducted. Contents of the counters 41 and 42 are evacuated into a stack region of the *RAM* (6) in a case where a command of sub-routine is conducted, and an interrupt is produced, or after an interruption command of a software is conducted.

(d) *Stack Pointer* (43)

The *Stack Pointer* (43) designates lower 8 bits of the highest address on a stack region of the *RAM* (6), and is decremented after the pushing of data into the stack region and incremented before the pulling of the data from the stack region. For instance, two hundreds fifty-six (256) bytes of addresses *2100* to *21FF* are allocated to the stack region in a logical address.

(e) *source high register* (45), *destination high register* (46), and *length high register* (47)

These registers function in case of a command of a block transfer. The *source high register* (45) provides an upper byte of a source address to designate the source address together with a content of the *X register* (39). The *destination high register* (46) provides an upper byte of a destination address to designate the destination address together with a content of the *Y register* (40). The *length high register* (47) provides upper 8 bits for a down counter together with a content of the *Accumulator* (38) so that a length of a block transfer is counted by a byte unit.

(6) Mapping Register 25 The *Mapping Register* (25) is composed of 8 registers each being of 8 bits to convert a logical address of 16 bits to a physical address of 21 bits, and is selected by upper 3 bits of the *H-bus* (35).

(7) Chip Enable Decoder 26 The *chip enable decoder* (26) provides chip enable outputs for following peripheral circuits by decoding upper 11 bits of a physical address.

(a) a chip enable for the *RAM* (6) . . . $\overline{CER}$

(b) a chip enable for the *Video Display Controller* (2) . . . $\overline{CE7}$

(c) a chip enable for the *Video Color Encoder* (3) . . . $\overline{CEK}$

(d) a chip enable for the programable sound generator 4 . . . $\overline{CEP}$

(e) a chip enable for the *Timer* (29) . . . $\overline{CET}$

(f) a chip enable for the input and output port . . . $\overline{CEIO}$

(g) a chip enable for the *interrupt request register* (30) and the *interrupt disable register* (31) . . . $\overline{CECG}$

(8) Timing and Control Unit 27 The unit 27 is connected to following terminals.

(a) $\overline{RD}$ terminal

A read timing signal is supplied through the $\overline{RD}$ terminal at a reading cycle.

(b) $\overline{WR}$ terminal

A write timing signal is supplied through the $\overline{WR}$ terminal at a writing cycle.

(c) SYNC terminal

A synchronous signal of *High* is supplied through the SYNC terminal at an instruction fetch cycle, that of *Low* is supplied therethrough at a system reset timing.

(d) $\overline{NMI}$ terminal

A non-maskable interrupt is produced when $\overline{NMI}$ input signal is supplied through the $\overline{NMI}$ terminal. A sub-routine call is conducted by reading lower address from the logic address *FFFC* and upper address from the logical address *FFFD* when a command which is conducted in a program is completed.

(e) $\overline{IRQ1}$ and $\overline{IRQ2}$ terminals

A sub-routine call is conducted by reading lower address from the logical 1 address *FFF8* and upper address from the logical address *FFF9* when $\overline{IRQ1}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

A sub-routine call is conducted by reading lower address from the logical address *FFF6* and upper address from the logical address *FFF7* when $\overline{IRQ2}$ input becomes *Low* in a case where a corresponding bit in the *interrupt disable register* (31) is *0*, and a corresponding bit in the *Status Register* (44) is *0*. At this time, the corresponding bit is set in the *Status Register* (44), and other corresponding bits are reset therein.

(f) REST terminal

A program is started by reading lower address from the physical address *001FFE* and upper address from the physical address *001FFF* when a $\overline{RESET}$ input becomes *Low*.

(g) $\overline{RDY}$ terminal

The CPU1 is started to operate when a $\overline{RDY}$ input is changed from *Low* to *High*.

(h) SX terminal

A complementary signal of a system clock signal is supplied through the SX terminal.

(i) OSCI terminal

An external clock signal is input through the OSCI terminal.

(j) $\overline{EA1}$ to $\overline{EA3}$ terminals

These are input terminals for a test of the CPU1.

(k) HSM terminal

A speed signal of *High* is supplied through the HSM terminal in case of a high speed mode of 21.477270 MHz/3, and that of *Low* is supplied therethrough in case of a low speed mode of 21.477270 MHz/12.

(9) Input and Output Port 28 The *input and output port* (28) is connected to following terminals.

(a) K0 to K7 terminals

The terminals are input ports from which data are written in accordance with the conduct of a reading cycle in regard to the physical addresses *1FF000* to *1FF3FF*.

(b) (00) to 07 terminals

The terminals are output ports with latches to which data are supplied in accordance with the conduct of a writing cycle in regard to the physical addresses *1FF000* to *1FF3FF*.

(10) Timer 29 The *Timer* (29) is connected to a test input terminal EAT for the CPU1 and provides a timer signal through the U-bus thereto.

(11) Interrupt Request Register 30 The register 30 is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *1* to show the $\overline{IRQ1}$ and $\overline{IRQ2}$terminals *Low* and the remaining 1 bit is *1* to show a timer interrupt caused. The register 30 is only used for *read*.

(12) Interrupt Disable Register 31 The register 31 is of 8 bits among which 5 bits are not used, while the remaining 2 bits are *0* to make an interrupt request of the $\overline{IRQ1}$ and $\overline{IRQ2}$ terminals disable, and the remaining one is *0* to make an interrupt request disable in accordance with the timer interrupt signal.

Next, operation of an apparatus for controlling a signal processing system to operate in high and low speed modes in the embodiment will be explained.

(1) High Speed Mode (HSM) When external clock signals of 21.477270 MHz is supplied through the terminal OSCI to the *timing and control unit* (27) as shown in [Figure 2], the *CPU* (1) interprets a program stored in *ROM* (5) thereby determining a high speed mode to be selected so that a command of $\overline{CSH}$ (Change Speed High) is conducted to set a high speed mode. In the high speed mode, system clock signals S_1, S_2, and S_3 (7.15909 MHz) as shown in [Figure 3] are produced by dividing the external clock signals of 21.477270 MHz by one-third. Simultaneously, a high speed mode signal *1* is supplied through the terminal HSM to a peripheral circuit. As a result, the *Video Display Controller* (2), the *Video Color Encoder* (3), the programmable sound generator 4 and so on which are connected to the *CPU* (1) are controlled to operate in the high speed mode. The system clock signals S_1, S_2, and S_3 are supplied as complementary signals through the terminal SX to a periphery circuit wherein one bus cycle is defined as a period from a rising edge of the system clock signal S_1, to a next rising edge thereof. The one bus cycle is a basic bus cycle by which a command conduct cycle is defined in each of commands.

(2) Low Speed Mode (LSM) External clock signals of 21.477270 MHz which are supplied through the terminal OSCI in accordance with a command CSL (Change Speed Low) are divided by one-twelfth. As a result, system clock signals (1.7897725 MHz) are produced as shown in [Figure 4], and supplied as complementary signals through the terminal SX to a peripheral circuit. Simultaneously, a low speed mode signal *0* is supplied through the terminal HSM to peripheral circuit.

(3) Change Over Between Two Modes

[Figure 5] shows a change over between high and low speed modes. Here, it is assumed that a command CSH by which a processing speed of the *CPU* (1) is set as a high speed mode, a non-operation command NOP, and a command CSL by which a processing speed of the *CPU* (1) is set as a low speed mode are produced at timings shown therein. A starting time at which the command $\overline{CSH}$ is produced is in a low speed mode during which system clock signals S_1, S_2 and S_3 are of 1.7897725 MHz in accordance with one-twelfth frequency division and supplied as complementary signals through the terminal SX. At this time, a mode signal HSM is *0*. After one bus cycle is then elapsed, a high speed mode is realized in accordance with the command CSH so that system clock signals are of 7.15909 MHz in accordance with one-third frequency division. At this time, a mode signal HSM is *1*. Next, when a command CSL is produced, a low speed mode is then realized after one cycle bus thereof. During operations, a program is counted by a *Program Counter* (PC).

Although the invention has been described with respect to specific embodiment for complete and clear disclosure, the appended claims are not to thus limited but are to be construed as embodying all modification and alternative constructions that may occur to one skilled in the art which fairly fall within the basic teaching herein set forth.

Category: Pages with broken file links

- This page was last modified on 7 November 2015, at 04:00.