PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G11B 20/00, 19/02, 20/18, 20/10

A1

(11) International Publication Number: **WO 98/43243**

(43) International Publication Date: 1 October 1998 (01.10.98)

(21) International Application Number: PCT/GB98/00747

(22) International Filing Date: 24 March 1998 (24.03.98)

(30) Priority Data:
9706198.0 25 March 1997 (25.03.97) GB

(71) Applicant *(for all designated States except US)*: MSU (UK) LIMITED [GB/GB]; Elder House, 526–528 Elder Gate, Central Milton Keynes, Bucks MK9 1LR (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: HOLLOWAY, Wynford [GB/GB]; Thurlstones House, Backside Lane, Sibford Gower, Banbury, Oxon (GB). BRENNAN, Martin [GB/GB]; Harraton House, 6 Church Lane, Exning, Newmarket, Suffolk CB8 7HF (GB).

(74) Agent: RAYNOR, Simon, Mark; Urquhart–Dykes & Lord, Midsummer House, 411C Midsummer Boulevard, Central Milton Keynes, Bucks MK9 3BN (GB).

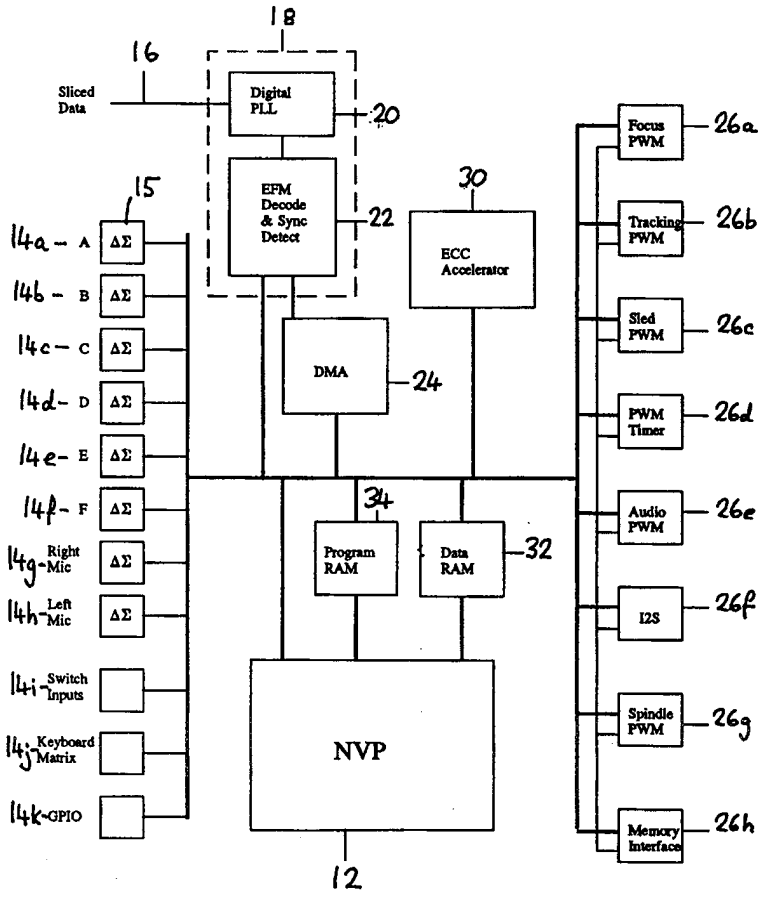
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: INTEGRATED CONTROL AND DECODING CIRCUITS FOR COMPACT DISK PLAYERS

(57) Abstract

An integrated control and decoding circuit for a CD player includes digital input means (14) for receiving input data read from the compact disk and control signals for controlling operation of the CD player. A RISC processor (12) serves both as a signal processor means for decoding and error correcting the input data and as a control means for controlling operation of the player. Digital output means (26) provide a plurality of digital output signals containing output data and control signals for controlling operation of the CD player. The digital input means (14), processor (12) and digital output means (26) are integrated onto a single semiconductor integrated circuit chip.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

INTEGRATED CONTROL AND DECODING CIRCUITS FOR COMPACT DISK PLAYERS

The present invention relates to an integrated control and decoding circuit for a compact disk player, and to a compact disk player including an integrated control and decoding circuit. In particular, but not exclusively, the invention relates to a high performance, low cost integrated circuit designed for video and audio compact disk (CD) players.

A compact disk player normally includes a laser light source, a set of four, five or six photo diodes that detect the light reflected from the compact disk, a set of motors or actuators for driving various components of the player and an electronic circuit for controlling operation of the above-mentioned components and decoding the digital signal received from the compact disk and converting it into either an analogue audio signal (in the case of an audio CD player) or a video signal (in the case of a video CD drive). Other signals may also be provided for use in different applications (for example, CD Karaoke equipment).

The electronics within an audio compact disk player normally consists of three separate components: the servo processor, which controls the drive motors of the CD player; the digital signal processor, which decodes the digital data representing the audio signal, and the micro controller, which controls and co-ordinates operation of the servo processor and the digital signal processor. The resulting circuit has a large number of chips and consequently is expensive, unreliable and inefficient. Previous CD players have also been inefficient in their error correction procedures.

It is an object of the present invention to mitigate at least some of the afore-mentioned disadvantages and to provide a general improvement in the cost and ease of manufacture of audio and video CD players.

According to the present invention there is provided an integrated control and decoding circuit for a CD player, said circuit including digital input means for receiving a plurality of digital input signals including an input data signal containing input data read from a compact disk playing in said CD player and an input control signal containing instructions for controlling operation of the CD player; signal processor means for decoding and error correcting said input data; control means for controlling operation of said player in response to said input control signal; and digital output means for providing a plurality of digital output signals including an output data signal containing output data derived from said decoded and error corrected input data and an output control signal containing instructions for controlling operation of the CD player; wherein said digital input means, signal processor means, control means and digital output means comprise digital electronic components that are integrated onto a single semiconductor integrated circuit chip.

By integrating the digital signal processor, the output means and the control means onto a single semiconductor substrate, the number of chips contained in the CD controller is drastically reduced. Manufacturing costs are thereby reduced and efficiency and reliability are increased.

The controller/decoder chip is preferably an all-digital component. This allows it to be built readily into many types of CD players, for example CD audio equipment, video disk players and CD Karaoke apparatus.

The integrated control and decoding circuit may include an analogue input interface having a plurality of analogue input ports for receiving analogue input signals and a plurality of analogue-to-digital converters (ADCs) connected to said analogue input ports, the digital output terminals of said ADCs being connected to said digital input means. The ADCs may be simple and relatively crude devices, for example Sigma-delta ADCs. Such converters are very cheap and reliable. Software can remove any offsets and provide compensation for gain. Said analogue input interface may be integrated onto said single semiconductor integrated circuit chip. Alternatively, a separate analogue input interface may be provided.

Advantageously, at least one digital output signal of said digital output means is modulated to provide an analogue drive signal. Said modulated output signal may comprise at least one digital output data signal and/or at least one digital output control signal. The modulated output signal may used to drive an analogue component such as an analogue amplifier directly.

Advantageously, said signal processor means comprises a RISC processor that decodes and error corrects said input data under software control. The integrated control and decoding circuit preferably includes a hardware error checking and correction accelerator for calculating the syndrome of a block of data and correcting the data using the value of the syndrome. Preferably, the syndrome is calculated by Galois field arithmetic, the arithmetic process being conducted by a plurality of XOR gates in combination with look-up tables.

The circuit provides for improved error detection and correction. The error detection and correction method used is preferably that known as CIRC (cross-interleaved Read Solomon code) and this process is carried out by a combination of software together with hardware that is specifically designed to provide support for the software. Although software error detection and correction processes are normally slower than dedicated hardware solutions, using specialised processor instructions, which have dedicated error detection and correction functions, to calculate the syndrome permits the error correction process to be carried out using one processor cycle per data word to accumulate the syndrome. Using a purely software solution requires a greatly increased number of instructions, which takes up a large amount of the computing power of the processor.

The speed of the Envoy error detection process is thus limited only by memory bandwidth and is as fast as a hardware solution, while maintaining the flexibility of a software approach. Adding dedicated error correction instructions to the Envoy allows error correction to be performed at a much higher rate, while still being under the control of software. The error correction protocol can thus be updated easily, for example when accepted standards are amended.

Advantageously, the circuit includes a digital Phase Locked Loop (PLL) to lock to the CD data stream.

According to a further aspect of the present invention there is provided a CD player comprising a CD drive, a CD pickup assembly and an integrated control and decoding circuit, said control and decoding circuit including digital input means for receiving a plurality of digital input signals including an input data signal containing input data read from a compact disk playing in said CD player by said CD pickup assembly and an input control signal containing instructions for controlling operation of the CD player; signal processor means for decoding and error correcting said input data; control means for controlling operation of said player in response to said input control signal; and digital output means for providing a plurality of digital output signals including an output data signal containing output data derived from said decoded and error corrected input data and an output control signal containing instructions for controlling operation of the CD drive and pickup assembly; wherein said digital input means, signal processor means, control means and digital output means comprise digital integrated components that are integrated onto a single semiconductor integrated circuit chip.

Advantageously, the CD player includes an analogue input interface having a plurality of analogue input ports for receiving analogue input signals and a plurality of analogue-to-digital converters (ADCs) connected to said analogue input ports, the digital output terminals of said ADCs being connected to said digital input means.

According to a preferred embodiment of the present invention, there is provided a single chip CD controller and decoder, which will be referred to hereafter as the "Envoy".

The Envoy contains all the digital logic required to control a CD mechanism, the CD player control buttons and display, and to decode and error correct a CD data stream, in a single semiconductor integrated circuit. This high level of integration reduces the cost of the final system by reducing the cost of components, reducing assembly time and increasing reliability because of the reduced component count. The Envoy in itself is a low cost part. It achieves this low cost both by using circuitry tailored specifically for this application and by using a high performance RISC processor, with an instruction set enhanced for CD error detection and correction, which allows the error correction to be performed under software control. This allows the RISC processor to be shared by both the CD mechanism control and error correction, giving flexible error recovery strategies with a lower logic gate count than using hardware error correction.

Various features of the Envoy are summarised below:

1. The Envoy is a highly integrated design that reduces the number of electronic components required to manufacture a CD player. It includes digital support for Delta Sigma Analogue to Digital Converters (ADCs), used to read the CD mechanism diode outputs, CD player button and display interfaces, an all digital Phase Locked Loop (PLL) to lock to the CD data stream, a RISC processor with hardware error correction support to decode the CD data, and single bit DACs to control the CD mechanism motors. The Envoy includes two I2S serial interfaces for connection to audio Digital to Analogue Converters (DACs) or MPEG video decoders. This level of integration reduces the overall cost of the CD player.

2. The envoy is a highly optimised design, using a RISC processor to decode the CD data stream and to control the CD mechanism. The CD data read from a CD will contain a large number of errors. Unless these errors are corrected, the CD data will not be usable. The CD data is recorded with extra information that allows corrupt data to be corrected. This method of encoding the data is called CIRC (cross-interleaved Read Solomon code) encoding. Decoding and error correcting data from the CD requires a large number of arithmetic operations on the data stream. The Envoy performs this process using the RISC processor (referred to hereafter as the "NVP"), which is controlled by software. Although software error detection and correction processes are normally slower than dedicated hardware solutions, using specialised processor instructions, which have dedicated error detection and correction functions, to calculate the syndrome permits the error correction process to be carried out using one processor cycle per data word to accumulate the syndrome. Using a purely software solution requires a greatly increased number of instructions, which takes up a large amount of the computing power of the processor. The speed of the Envoy error detection process is thus limited only by memory bandwidth and is as fast as a hardware solution, while maintaining the flexibility of a software approach. Adding dedicated error correction instructions to the Envoy allows error correction to be performed at a much higher rate, while still being under the control of software. The error correction protocol can thus be updated easily, for example when accepted standards are amended.

3. The Envoy NVP has an enhanced instruction set that supports the Galois field arithmetic required for correcting Reed-Solomon encoded data streams.

4. The design is highly optimised for a CD controller, with the balance between hardware, software and memory for software storage arranged so that each performs its task without creating a processing bottleneck. The Envoy makes very effective use of logic gates to reduce the component cost.

5. As the Envoy is programmable, it can be modified for special requirements and adapted for any CD mechanism by altering the software. Envoy has sufficiently flexible I/O that it can interface directly to liquid crystal displays, push button switches and an infrared remote control. On-chip analogue outputs can be used directly for less demanding medium quality audio applications and the I2S interface can be used to drive external DACs for top audio quality. There is sufficient spare analogue I/O and processing bandwidth to be used for Karaoke effects processing. If software customisation is not sufficient, the logic may be modified to create a new version of the Envoy. The Envoy is designed without custom logic blocks so it can be easily modified and can be transferred between integrated circuit manufacturers relatively easily.

6. The Envoy downloads its program on power-up. This may either be from a ROM or via a system control processor. Modifications and customisation only require ROM changes. Different designs may use one printed circuit board and be customised by fitting the required ROM on final assembly to reduce the number of types of different types of printed circuit boards required to be held in stock.

7. Complex algorithms may be used to detect and correct CD errors. These algorithms may require too much hardware to be possible from a hardware error corrector. Algorithms may be updated throughout the life of the product.

8. The Envoy is programmable and this allows it to he built readily into many types of CD players, for example CD audio equipment, video disk players and CD Karaoke apparatus.

9. The Envoy has an interface to DRAM, allowing large amounts of data to be stored for effects such as reverberation as used for Karaoke.

10. When connecting new mechanisms, the Envoy can be programmed to pass debugging information on the state of the servo loops and the error correction process to a PC to allow the system to be optimised. Because these features are under software control, their state is accessible for debugging.

11. The Envoy is connected to the analogue outputs of the photodiodes in the CD pickup (and any other analogue input devices) by means of analogue-to-digital converters (ADCs). As the Envoy is programmable, these may be simple and relatively crude devices, for example Sigma-delta ADCs. Such converters are very cheap and reliable. The software is able to remove any offsets and provide compensation for gain changes.

An embodiment of the invention will now be described, by way of example, with reference to the accompanying drawings, in which:

Fig. 1 is a diagram of the Envoy integrated circuit, showing the pin inputs and outputs;

Fig. 2 is a block diagram of a CD player including the Envoy integrated circuit;

Fig. 3 is a block diagram illustrating the architecture of the Envoy integrated circuit;

Fig. 4 is a timing diagram of the Philips*, Sony* and Matsushita* CD data formats;

Fig. 5 is a timing diagram for a non-multiplexed LCD drive;

Fig. 6 is a timing diagram for a 3-way multiplexed LCD drive;

Fig. 7 is a simplified circuit diagram of a sigma delta analogue to digital converter;

Fig. 8 is a memory map of the Envoy integrated circuit;

Fig. 9 is a timing diagram for the fast DRAM RAS (dynamic random access memory row address strobe) transfer followed by the page mode cycle;

Fig. 10 is a timing diagram for the slow DRAM RAS transfer followed by the page mode cycle;

Fig. 11 is a timing diagram showing the relative timings of the fast and slow RAM memory cycles;

Fig. 12 is a block diagram illustrating the architecture of the NVP processor;

Fig. 13 is a simplified circuit diagram of the PLL (phase locked loop) circuit;

Fig. 14 is a simplified diagram illustrating operation of the scrambler;

Fig. 15 is a flowchart of the C1 error detection and correction process, and

Fig. 16 is a flowchart of the C2 error detection and correction process.

*Philips, Sony and Matsushita are acknowledged to be Trade Marks.

The Envoy 1 is a single integrated circuit, supplied for example as a 100 pin plastic quad flat pack (PQFP) package. The Envoy is available as a die for mounting directly on a Printed Circuit Board (PCB). The pinout is shown in fig. 1, and the functions of the pins is defined in table 1 below.

Pin	Input/Output	Function	
clk	Input	Main clock input.	
resetl	Input	Active low reset input.	
slice	Input	CD data input. An external circuit compares the analogue signal from the central photo diodes against a DC level to produce a digital wave form with as many ones as zeros on average.	
di0-di3	Input	Photo diode inputs. These four inputs are associated with ADC channels 1 to 4. These inputs expect the photo diode outputs to have been digitised using sigma delta converters which will produce a stream of ones and zeros. Internal logic converts the bitstream to a value representing the output each diode by counting the number of ones in each input during each sample rate window.	
irin	Input	Infra Red control input	
trk	4mA Output	Tracking coil PWM output.	
foc	4mA Output	Focus coil PWM output.	
spp, spm	4mA Outputs	Bridged spindle motor PWM outputs.	
slp, slm	4mA Outputs	Bridged sled motor PWM outputs.	
wel	8mA Outputs	Data RAM write enable.	
da0-da15	4mA Outputs	External data ram address output. Note: these outputs are used as inputs during reset to configure the Envoy as follows:	
		da0	TMODE
		da1	BOOT from 8 bit ROM
		da2-4	Slow ROM speed
		da5	DRAM speed
		da6-7	DRAM columns

Pin	Input/Output	Function	
		da8	ROM test mode
		da9	Use separate CD system clock
		da10	Not Used
		da11	Memory Test Mode
		da12	Not Used
		da13	Boot from a serial link
dcsl	8mA Output	Quick chip select. This active low signal is asserted for data memory in the addresses range 1M-2M. Memory in this region has a cycle time of one clock cycle. This is intended for external RAM to extend the internal data memory.	
scsl	8mA Output	Slow chip select. This active low signal is asserted for data memory in the addresses range 3M-4M. Memory in this region has a cycle time between two and nine clock cycles. This area of memory is intended for ROM.	
oel	8mA Output	Data RAM output enable. This active low signal enables the data outputs on data RAM chips.	
dd0-dd7	4mA In/Out	Data RAM data bus.	
v0- v31	4mA In/Out	General Purpose I/O. Each of these pins may be used by the processor as an input or an output. Many of the pins have alternate functions as described later.	
clkout	4mA Output	Half clock speed output	
error flag	4mA Output	Software controlled error pin.	

Table 1 The Pin Description

pin	function	pin	function	pin	function	pin	function
1	5V	26	da3	51	5V	76	5V
2	gnd	27	+5V	52	gnd	77	gnd
3	clk	28	gnd0	53	v6	78	v25
4	resetl	29	da4	54	v7	79	v26
5	slice	30	da5	55	v8	80	v27
6	di0	31	da6	56	v9	81	v28
7	di1	32	da7	57	v10	82	v29
8	di2	33	da8	58	v11	83	v30
9	di3	34	da9	59	+5 V	84	v31
10	irin	35	5V	60	gnd	85	5V
11	5V	36	gnd	61	v12	86	gnd
12	gnd	37	da10	62	v13	87	foc
13	dd0	38	da11	63	v14	88	trk
14	dd1	39	da12	64	v15	89	spp

pin	function	pin	function	pin	function	pin	function
15	dd2	40	da13	65	v16	90	spm
16	dd3	41	da14	66	v17	91	slp
17	dd4	42	da15	67	3.3 V	92	slm
18	dd5	43	3.3V	68	gnd	93	3.3V
19	3.3V	44	gnd	69	v18	94	gnd
20	gnd	45	v0	70	v19	95	scsl
21	dd6	46	v1	71	v20	96	dcsl
22	dd7	47	v2	72	v21	97	wel
23	da0	48	v3	73	v22	98	oel
24	da1	49	v4	74	v23	99	clkout
25	da2	50	v5	75	v24	100	errorflag

Table 2 The Envoy Pinout

1. THE ENVOY AS PART OF A CD PLAYER

The Envoy 1 is designed to be the main electronic component of a CD player, a block circuit diagram of which is shown in Fig. 2. The Envoy 1 interfaces to the CD photodiodes 2 via an analogue front end 3, which may consist of discrete components or an integrated circuit. This analogue front end 3 contains ADCs which convert the photodiode outputs to digital signals for the Envoy. The Envoy 1 connects directly to the CD control buttons 4 and drives the motors 5 of the CD mechanism and a display 6, which may be either an LCD or a LED display. Optional devices are high quality audio DACs 7 to deliver high fidelity sound, a video decoder 8 for video CD players and a control microprocessor 9, which may take control of the CD player. The Envoy 1 downloads its program either from a ROM 10 or via a serial link from the microprocessor 9.

2. THE ENVOY ARCHITECTURE

At the heart of the Envoy chip is a specialised RISC processor 12, which will be referred to hereafter as the "NVP" processor. The NVP processor 12 is augmented with peripheral logic, some of which is general in nature and some of which is specific to the task of CD control and playback. The ADC inputs, DAC outputs, timers, flexible I/O, interrupts, DMA channels and memory controller are general in nature and can be used in a variety of embedded applications. The digital phase locked loop, data separator and ECC logic are specific to CD. The system architecture of the Envoy chip will now be described with reference to Fig. 3.

The NVP processor 12 is connected to a plurality of input ports 14, from which it receives digital input data. The input ports 14 may include, for example, six ports 14a-14f for receiving input signals from the photo diodes of the CD pickup assembly, right and left microphone ports 14g-14h, a switch port 14i, a keyboard matrix port 14j and a general purpose input/output (GPIO) port 14k.

Where the input device provides an analogue output signal, as for example in the case of the photo diodes of the CD pickup assembly, that device may be connected to the relevant input port 14 via an analogue-to-digital converter (ADC) 15. This may, for example, be a Sigma-Delta ADC (shown in more detail in Fig. 7). Where the input device provides a digital output signal, as for example in the case of the switch inputs, that device may be connected direct to the relevant input port 14.

The digitally-converted signals received from the photo diodes through ports 14a-14f are used by the NVP processor 12 to control the focus and tracking of the CD pickup, via output ports 26a, 26b.

The NVP processor 12 also receives sliced data from the photo diodes, which indicates the presence or absence of a pit at the focus of the laser. This signal represents the data carried by the compact disk and is derived from the sum of the currents from the central one, two or four photo diodes. This signal is referred to as the central aperture signal and contains frequencies in the 200-700kHz band. The CD encoding scheme balances the number of ones and zeroes in the data. The Envoy uses an analogue to digital converter with an AC coupled amplifier to turn the high frequency signal into a digital signal while ignoring slower variations in amplitude or offset caused by scratches or dirt on the disk surface.

The sliced data is received through input port 16. Optionally, the Envoy chip 1 may use a data separator 18, which processes the sliced data before it reaches the NVP processor 12. The data separator 18 includes a digital phase locked loop (DPLL) 20 and an EFM decode and synchronisation detector 22. The data separator 18 is bypassed for DVD players and an external data separator is used. A direct memory access (DMA) controller 24 is connected to the EFM decode and synchronisation detector 22. Operation of the data separator 18 is described in more detail below.

On the output side of the Envoy chip there are provided a plurality of output ports 26, including pulse width modulated (PWM) output ports for controlling the focus, tracking, sled, and spindle actuators/motors (26a,b,c and g), as well as the timer and audio output (26d and 26e). Other output ports are provided for the I2S and memory interfaces (26f and 26h). Where an analogue output is required, the relevant output port may he connected to a suitable digital-to-analogue converter (DAC).

The motors/actuators connected to output ports 26a,b,c,g may be driven by pulse width modulated (PWM) or pulse density modulated (PDM) outputs. Two types are used. Unipolar outputs drive a motor in one direction while bipolar outputs allow the motor direction to be reversed. Bipolar motor drives use a bridge circuit, which allows a motor to be driven forwards and backwards from a single supply. This in turn requires two digital outputs, one of which pulses high for positive (forward) values whilst the other pulses high for negative (backward) values. The pulses are generated at the audio sample rate and the width of the pulses determines the speed of the motor. The precision is determined by the ratio between the clock frequency and the sample rate, faster clocks providing greater precision.

The focus and tracking actuators connected to output ports 26a and 26b are driven by linear amplifiers that amplify a filtered digital PDM signal. The PDM signal contains a pseudo

random bit stream in which the number of ones in the sample period represents the value of the output and gives a modified frequency spectrum to PWM outputs.

Also included within the Envoy architecture are an error checking and correction (ECC) mechanism accelerator 30, data random access memory (RAM) 32 and program RAM 34.

The error checking and correction accelerator 30 calculates the syndrome of a block of data. The syndrome comprises four bytes that are zero if there is no error and whose values can be used to identify and correct errors. The syndrome is first initialised to zero then each byte in the block is written to a register. Writing the byte causes each byte of the syndrome to be multiplied by increasing powers of a constant and XOR'd with the byte. When all the bytes have been written the syndrome bytes can be read. The error correction logic replaces the twenty four instructions that would previously have been required with just one. The multiplication process, which is Galois field multiplication rather than arithmetic multiplication, is conducted by a number of XOR gates.

The laser in the CD pickup assembly is driven by a current source which is in turn driven by a filtered PWM signal. The output of the laser is monitored by a photo diode within the laser package. Most CD pickups are trimmed so that the laser power management (LPM) diode produces a specific voltage for correct laser output. Such pickups are therefore self-regulating. Alternatively, however, the output of the LPM diode can be monitored by the Envoy chip and the laser output regulated accordingly.

3 ENVOY INTERFACES

3.1 I2S INTERFACE - AUDIO AND VIDEO

The I2S interface 26f is a bi-directional interface, used to output audio data to high performance audio DACs, or a video stream to a video decoder in a video CD player. The I2S interface may be programmed to input serial data from another device.

There are three main data formats, Philips, Sony and Matsushita, as shown in Fig. 4.

3.2 GENERAL PURPOSE INTERFACE

The Envoy 1 has 32 general purpose digital input/output pins which may be individually selected as input or output. These may be used for CD button inputs and display outputs. In addition, some of the pins have a dedicated use, selected by software, detailed below:

Pin	Input	Output
v0	External data separator clock input	data memory address a16
v1		data memory address a17
v2		data memory address a18
v3		data memory address a18
v4	ADC4	Recovered CD clock
v5	ADC5	Recovered CD data

Pin	Input	Output
v6	ADC6 and Serial Boot Clock	
v7	ADC7 and Serial Boot Data	
v8	Serial Boot Control 0	DAC4
v9	Serial Boot Control 1	DAC5
v10		DAC6
v11		DAC7
v12		IODEC1
v13		IODEC2
v14		IODEC3
v15		IODEC4
v16		RASL
v17		CASL
v18	IRQ4 Input	CD sample rate interrupt
v19		
v20		
v21		
v22		EXT DMACK
v23	I2S 2 RX	
v24		
v25	I2S 1 CLK	I2S 1 CLK
v26		I2S 1 WS
v27	I2S 2 CLK	I2S 2 CLK
v28		I2S 2 WS
v29		I2S 2 TX
v30	I2S 1 RX	
v31		I2S 1 TX

Driving LCDs requires further description. A non-multiplexed LCD has one common back plane and one electrode for each display element. The back plane should be driven with a low frequency square wave with a 50% duty cycle. The display elements should be driven with the same square wave in-phase for display elements which are off and out-of-phase for display elements which are on. This system drives the "on" elements with the maximum AC voltage with no DC component (which would otherwise damage the display). "Off" elements receive no drive at all. The relative timings of a non-multiplexed LCD drive are shown in Fig. 5.

A multiplexed LCD has a small number (three or four) of back planes or rows and a larger number of electrodes or columns. Display elements are at the intersection of rows and columns.

Each of the rows are driven in turn. When a row is driven it is driven high for a time then low for an equal time. When the row is not driven the output should be disabled (tri-stated). Two external resistors bring the voltage on the row exactly half way between the supplies when the row is undriven.

The columns are driven in several phases corresponding to each of the rows. If the display element at the intersection should be off then the column is driven in-phase with the row. If the display element is on then it is driven out-of-phase with the row. The relative timings of a 3-way multiplexed LCD drive are shown in Fig. 6.

When a row is active the display elements are driven with either 5 volts or zero depending on whether they are on or off. When a row is inactive the display element is driven with 2.5 volts regardless of the state of other elements on the same column.

The system breaks down if the display has a threshold voltage very different from 2.5V. In this case the drive voltage can be increased or decreased by adding an extra phase during which all the rows are driven together. All the columns are then driven in-phase to reduce the average drive voltage or out-of-phase to increase the average drive voltage. This adjusts the threshold at the expense of the on/off ratio.

3.3 ANALOGUE INPUTS

Envoy has eight digital inputs 14a-14h which, combined with external 1 bit delta sigma converters 15, are used as analogue to digital converters. A simplified view of a Sigma Delta ADC 15 is shown in Fig. 7. The external Sigma-delta converter uses an integrator 40, a comparator 42, a latch 44, a one bit DAC 46 and an amplifier 48 to convert an analogue input voltage to an output stream of digital ones and zeros whose average value is proportional to the analogue voltage. The voltage is digitised by counting the ones in a given interval.

The output of the sigma delta converter 15 is input into the Envoy 1, and the number of ones is counted in a sample window, to give a numerical value equivalent to the analogue input value at the front end of the converter. This value may be read from the ADC register. The absolute value read from the converter voltage will depend on the gain and offsets introduced by the analogue components. However, as the Envoy uses software to control the servo loops in the CD system, the software may compensate for gain and offset variations.

3.4 ANALOGUE OUTPUTS

The Envoy has eight digital outputs 26a-26h which are pulse width modulated or pulse density modulated so that when the signal is filtered, these signals represent analogue drive voltages. These are used to control the CD mechanism motors and laser power. An outputs is also available as an audio output 26e for a medium quality CD player. There are two types of these outputs, unipolar and bipolar. Both types have hardware to generate a pulse width modulated (PWM) or pulse density modulated (PDM) output at a programmable sample rate. Unipolar outputs drive a single signal and are used to drive a motor in one direction. Bipolar outputs drive two signals for bridge driving a motor in two directions.

3.5 MICROPROCESSOR DIAGNOSTIC INTERFACE

The NVP 12 has a number of diagnostic registers which are used for testing the operation of the Envoy. These are activated by resetting the Envoy with PA0 pulled high, which will start the test mode, TMODE. This allows the registers to be accessed to start and stop NVP, set operating conditions and read and write to NVP memory and I/O.

The signals used for this access are as follows:

data 7:0	data 15:8	address	read strobe	write strobe
dd7..dd0	pa7..pa0	pa11..pa8	v27	v28

3.5.1 HOST REGISTER SUMMARY

Address	Read	Write
0		DIAGCMND
2	DIAGADDL	
4	DIAGPDL	
6	DIAGPDH	
8	DIAGMD	DIAGMD
A	DIAGLINK	DIAGLINK
C	DIAGADDH	

3.5.2 HOST REGISTER DETAILS

DIAGCMND	Diagnostic command register	0	16 bit W/O

Bit 0	nReset	This bit enables NVP program execution. When clear the test interface has access to NVP program and memory space.

DIAGADDL	Diagnostic address low register	2	16 bit W/O

This register is used for setting the high 14 bits of an NVP data memory address.

DIAGADDH	Diagnostic address high register	Ch	16 bit W/O

This register is used for setting the low 16 bits of an NVP data memory address.

DIAGPDL	Diagnostic program data low register	4	16 bit W/O

This register holds the bottom 16 bits of program data.

DIAGPDH	Diagnostic program data high register	6	16 bit W/O

This register holds the top 16 bits of program data.

DIAGMD	Diagnostic memory data register	8	16 bit R/W

This register holds 16 bits of data from memory.

DIAGLINK	Diagnostic link register	Ah	16 bit R/W

These 16 bit registers allows rapid bi-directional communication between the test interface and the NVP 12. Data written by the NVP may be read by the test interface and data written by the test interface may be read by NVP. Transfers from NVP to the test interface may be concurrent with transfers from the test interface to NVP.

3.6 MEMORY INTERFACE

The Envoy can run using its own internal program and data RAM 34,32. However, if more data storage is required, it can interface to external memory via memory interface 26h. This may be static or dynamic RAM 10 for extra data storage. This memory is especially useful for Karaoke echo effects. In addition, the Envoy will connect to 8 bit wide ROM to allow its program to be downloaded after it is reset. Out of the possible 1Gbyte of data memory (30 bit address) that the NVP 12 can address, only the first 16 Mbytes (24 bits) is decoded as illustrated in the memory map shown in Fig. 8.

Addresses above 1M byte are external memory. Addresses below 1M are reserved for internal memory although only a limited amount has been provided initially. Internal and external memory are handled separately.

Whatever page address is set, hardware in the NVP 12 maps the first 16k of memory into the first 16k of internal data memory. This area is intended for frequently accessed variables and smaller data structures, which always remain visible irrespective of changes to the page register. Memory mapped I/O occupies the top 256 locations of the bottom 16k so is always visible.

External memory is 8 bits wide. The external memory is split into four areas. From 0 to 1 Mbyte is allocated to fast static data ram. This must allow access within 1 processor clock cycle. Part of this area overlays the internal memory and this area is not accessible. This area of memory is selected by the signal dcsl.

Addresses from 3 to 4 Mbytes are allocated to ROM. The cycle time for this memory is between 2 and 9 clock cycles, set by configuration bits when the Envoy is reset. This area of memory is selected by the signal scsl.

Addresses from 2 to 3 Mbytes are used for slow RAM. The access time is the same as the ROM. The memory is split into four 256 byte areas, using signals vi12 to vi15 as chip select lines.

Addresses from 4 to 16 Mbytes are used for DRAM. The chip generates a multiplexed address.

3.6.1 MEMORY TIMING

Figs. 9 to 11 show the timing of various external memory cycles. Almost all signals are derived from the rising edge of the clock. The exceptions are the write strobes and the write data. For single cycle memory (program RAM and the first megabyte of external data RAM) the write strobe is derived from the low phase of the clock. For slower memory the write strobe is a whole number of clock cycles but is derived from the negative edge of the clock.

The write data is enabled by the write strobe.

In all cases read data is latched by the rising edge of the clock at the end of the transfer.

Fig. 9 illustrates the relative timings of the fast DRAM RAS (dynamic random access memory row address strobe) transfer followed by the page mode cycle, and shows RAS reads and writes and page mode reads and writes to fast DRAM. The DRAM speed is set by a configuration resistor during reset. The FAST/SLOW setting is determined by the system clock speed as well as the DRAM specification. Because the DRAM timing can only be adjusted in whole clock cycles there may be occasions where increasing the clock frequency decreases DRAM bandwidth because the SLOW setting must be used to stay within the DRAM specification. Fast DRAM has a six clock RAS transfer and a two clock page mode transfer.

Fig. 10 shows the relative timings of the slow DRAM RAS transfer followed by the page mode cycle. Slow DRAM has an eight clock RAS transfer and a three clock page mode transfer.

Fig. 11 shows the relative timings of the fast and slow static RAM memory cycles. The first megabyte of external memory has a one clock cycle access time. The next two megabytes have an access time between two and nine clock cycles.

4. ENVOY CONFIGURATION

The Envoy is configured in two ways. Firstly, it is configured by reading values on the RAM address lines after being reset. This sets the hardware options so that the Envoy can communicate with the rest of the system. The second part is loading the program into internal memory before running it. This happens after reset, and the method of loading the program is controlled by links read during the hardware setup phase.

4.1 HARDWARE CONFIGURATION

When the Envoy is reset, hardware links pull address lines high or low via 10KΩ resistors on the PCB. These pins have the following functions:

Pin	Function	Description
DA0	TMODE	This is used to test the Envoy. If this input is held high during reset, the Envoy will be set to the test mode. In this mode, the internal registers are visible via the data and address pins.

Pin	Function	Description			
DA1	BOOT FROM ROM	If pulled high then after reset NVP loads its program RAM from a byte wide ROM on the data bus.			
DA2-4	SPEED	These three inputs determine the cycle time of the memory or peripheral associated with the ROM chip select scsl. The cycle times are as follows:			
		PA4	PA3	PA2	Time
		0	0	0	2
		0	0	1	3
		0	1	0	4
		0	1	1	5
		1	0	0	6
		1	0	1	7
		1	1	0	8
		1	1	1	9
DA5	FAST	This input determines how many cycles are used in DRAM transfers. If low five cycles are taken by the row address phase and three cycles by the column address phase (page mode cycle time). If high four cycles are taken by the row address and two by the column address phases.			
DA6-7	COLS	These inputs specify the number of columns in the attached DRAM. This is used by the chip to correctly multiplex the address for DRAMs.			
		PA7	PA6	Columns	
		0	0	256	
		0	1	512	
		1	0	1024	
		1	1	2048	
DA8	QUICK	If this input is held high during reset, the Envoy will reduce the access time for external ROM.			
DA9	EX-CDCLK	If high this bit selects a second clock input for the entire CD subsystem. The second clock should be input on pin v21 and may be up to three times faster than the main (processor) clock.			
DA10		Not Used			
DA11	MTEST	If this input is held high during reset, the Envoy will enter its RAM test mode, and the RAM will be directly accessible from pins.			
DA12		Not Used			
DA13	SERIAL BOOT	If this input is held high during reset, the Envoy will load its program RAM from data entered via a serial port, before starting to execute its program.			

4.2 PROGRAM LOAD

The Envoy program RAM 34 is internal and must be loaded before the Envoy can run. This can happen in one of two ways, controlled by the state of pins PA1 and PA13 when the Envoy is reset. If PA1 is pulled high on reset, the Envoy will load its program from an external 8 bit ROM. If PA13 is pulled high on reset, the Envoy will receive its code via a serial port from the host processor. In both cases, the Envoy will start program execution at address 0 after the program load is complete.

When loading from a serial port, the program RAM to be loaded down a 4 wire serial interface using the following signals:

Input	Use
vi6	Serial Load Data, true sense, msb first
vi7	Serial Load Clock, clocking data on the rising edge
vi8	Serial Load Control 0
vi9	Serial Load Control 1

Commands use the two control lines as follows:

Control 1	Control 0	function	use
	0	Idle	No action - will not load data or an address
0	1	Load data	Data previously serially will be loaded into the current address and the address will be incremented
1	0	Load address	Data previously loaded serially will be used to set the current address
1	1	Shift in data	Accept serial data into the internal shift register

The interface can be used in the 3 wire mode if vi8 is pulled high with a resistor . This limits the commands that can be sent down the interface.

The load sequence may be terminated in one of two ways, firstly by writing 32K of data, in which case the boot will automatically stop, or by setting the input vi0 high, when the processor will start executing its program from address 0.

5. THE NVP RISC PROCESSOR

NVP 12 is a small fast embedded processor. It uses on-chip memory to perform one 32 bit op code every clock cycle. Its key features are as follows:

- Small, using about 12k gates, 1½k words of data RAM and 5k words of instruction RAM

- Fast - 33 MIPS
- 1Gb available address space, 16 Mbytes used
- 16 byte wide registers can be used in twos or fours
- 32 bit ALU
- All instructions take one cycle
- Nine interrupt sources
- One clock interrupt latency
- Prioritised and vectored interrupt response
- 32 bit op code
- One DMA channels
- Serial multiplier
- Indexed addressing carries no overhead
- Immediate data available every instruction
- Single cycle load & store to internal memory
- Harvard architecture overlaps program fetch with loads & stores
- Shifter shifts 32 bits left or right four bits per clock
- Modular addressing mode good for buffer management
- Conditional branches and calls
- Powerful error correction operations supporting Galois arithmetic
- Timer

5.1 THE NVP ARCHITECTURAL OVERVIEW

The NVP architecture is shown in Fig. 12. The NVP 12 has 16 eight bit registers 50 which are labelled R0 through to R15. These can be used individually, in pairs or in fours for byte, word and long word operations. When using words or long words the lowest register is used as the label. So R0, R2, R4, R6, R8, R10, R12 and R14 are word registers and R0, R4, R8 and R12 are long word registers.

The Registers 50 may be used as data or as addresses. ALU operations typically combine a source or immediate operand and destination register in one instruction. Memory operations typically use an address register as an index, an offset and a destination data register in one instruction.

There is a 14 bit program counter which can address programs up to 16k instructions in internal or external memory. NVP 12 supports conditional jumps and calls with constant or computed destinations. A call or interrupt causes the PC to be pushed onto the stack in data memory along with the two flags: carry and zero. This maximises subroutine and interrupt performance.

The stack can be placed anywhere in the first 64k of data memory and the stack pointer 54 allows calls and interrupts to be nested up to 64 deep. Stack based addressing is not supported because it can be supported by indexed addressing.

There are just two flags: carry and zero. Their meaning reflects the operand size of the relevant instruction. The conditional instructions JUMP and CALL use all meaningful combinations of these flags.

NVP has separate program and data memory and has a three stage pipeline which comprises of program fetch, execution and loads/stores. All stages execute in parallel. To simplify the design certain peculiarities of pipelined execution remain exposed to the programmer. This allows the canny programmer to exploit machine cycles which would otherwise be discarded. These peculiarities include:

- The instruction after a JUMP or CALL is always executed
- Two instructions after a RET are always executed.
- The target of a LOAD instruction is not affected (at least) until the end of the following instruction.

NVP 12 has an address space of 1Gb which can be accessed with 32 or 16 bit pointers. 32 bit registers address the whole of memory directly. 16 bit pointers may be direct (part of the op code) or indirect (register + offset). 16 bit pointers address two areas of memory at once. Addresses in the range 0-16k always address the first 16k of memory. This can be used for static system variables and for applications with small amounts of data. Addresses in the range 16k-64k are added to a page register to provide a 48k window which can be put on any 16k boundary in memory. This can be used for intermediate sized data which still has locality of reference e.g. CD-ROM error detection and correction.

A modular addressing mode can be used to contain pointers within a buffer. This is particularly useful for cross interleave error correction.

NVP has a 16 bit memory data bus. The ALU 56 can add, subtract and shift 32 bit data but loads and stores are restricted to bytes and words.

In addition to internal program memory 34 and data memory 32, the NVP may use external data memory 58. External memory is decoded into four banks:

- Quick static RAM (1M byte)
- Slow static RAM 1 (1M byte)
- Slow static RAM 2 (1M byte)
- Dynamic RAM (12M byte)

NVP 12 can transfer data to/from internal memory 32 while a transfer to/from slower external memory 58 is completing. The processor will wait for the external transfer to complete only if the WAITM instruction is executed or if a second external transfer is requested. Apart from these two instances all instructions execute in one clock cycle.

NVP has an interrupt controller 58 and nine interrupt sources which are independently maskable. Interrupts are prioritised and vectored to addresses 10h, 20h, 30h and so on. All interrupts are enabled or disabled with the disable interrupt (DI) and enable interrupt (EI) instructions. Interrupts are disabled on entry to the interrupt service routine and enabled on return from interrupt (RETI).

NVP has a DMA channel 24 which is normally used by the CD to put CD data into memory. The DMA channel uses unused memory "slots" which appear in the program so DMA has no impact on processor performance.

NVP uses a serial multiplier/accumulator 60. This uses the 32 bit adder in the ALU with a little extra logic to provide a multiply/accumulate function at very little cost. In essence the MACC instruction multiplies R0 by the bottom two bits of R14 and adds the product to the destination register. This process need only be repeated as warranted by the precision of R14. So if R14 is an eight bit number then four cycles are required to perform the multiply accumulate.

NVP has a 32 bit shifter 56 which can perform all the useful single bit shifts and rotates on 8, 16 and 32 bit operands. It can also perform arithmetic shifts of up to four bits per clock.

NVP has been designed to perform error checking and correction as fast as possible. The computation intensive syndrome calculation is limited only by the speed of the internal memory so can be calculated as fast in software on NVP as in dedicated hardware.

NVP uses a 16 bit counter 62 running off the system clock to allow more accurate measurements of intervals between transitions in external inputs. The general purpose inputs v46 and v47 generate interrupts on every transition. This could be used, for instance, to provide a simple IR remote control receiver.

5.2 NVP I/O MAP

All I/O is 16 bit and is mapped into addresses 3F00 to 3FFF

Address	Write	Read
3F00	CDCNTRL	CDSTAT
3F02	PLLFREQ	MAX
3F04	Focus Motor DAC1	Diode 0 ADC1
3F06	Tracking Motor DAC2	Diode 1 ADC2
3F08	Spindle Motor BDAC3	Diode 2 ADC3
3F0A	Sled Motor BDAC4	Diode 3 ADC4
3F0C	I2S1CNTRL	ADC5
3F0E	Laser Power DAC5	ADC6
3F10	Left Audio DAC6	ADC7
3F12	Right Audio DAC7	
3F14	SYNCLR	SYNLOW
3F16	CD Error	SYNHIGH
3F18	DAC8 ??	SUBCODE
3F1A	CDTIMER	PLL Read
3F1C	I2S1TXL	I2S1RXL
3F1E	I2S1TXR	I2S1RXR
3F20	OUT0	IN0
3F22	OUT1	IN1
3F24	OUTSEL0	
3F26	EN0	

Address	Write	Read
3F28	EN1	
3F2A	OUTSEL1	
3F2C	PLLCOEF	
3F2E	I2S2TXL	I2S2RXL
3F30	I2S2TXR	I2S2RXR
3F32	I2S2CNTRL	
3F34	IRT	
3F40	LINK	LINK
3F42	Program Address Low	
3F44	Program Data Low	
3F46	Program Data High	
3F48	Control Register	

5.3 NVP I/O IN DETAIL

CDCNTRL	CD control register	3F00h	Write only

Bit 0	PLLEN	This bit enables the CD Digital-Phase-locked-loop (DPLL) and data separator. The PLL automatically adjusts the recovered data clock to match the frequency and phase of the incoming sliced data.
Bit 1	DMAEN	This bit enables the Data DMA mechanism. Once the DPLL has acquired lock this bit is set and CD data is DMA'd into NVP memory starting with the first byte of the next frame.
Bit 4	EXTPLL	This bit configures the CD logic to work with an external Phase
Bit 5	RESYNC	This bit enables the re-synchronisation mechanism. When the mechanism is enabled a sync coincidence will reset the DMA mechanism no matter how far it had drifted.
Bit 6	WIDE	his bit widens the ENVOY1 sync window from +/-6 bits to about +/- 24 bits.
Bit 7	WIDE RESYNC WINDOW	This increases the width of the window which can be used to resync the PLL.
Bit 8	SYNCED1	Writing a one to this bit clears the SYNCED2 flag.
Bit 9	CLR-FREADY	Writing a one to this bit clears the frame ready flag in the CD status register
Bit 10	VEARLY ENABLE	This bit disables or enables the vearly mechanism.

Bit 11	VEARLY POSITION 0	These two bits control the value substituted for the phase detector output when vearly is active.
Bit 12	VEARLY POSITION 1	
Bit 13	VEARLY KICK 0	Vearly kick controls the size of the vearly signal used when the vearly mechanism is in effect.
Bit 14	VEARLY KICK 1	
Bit 15	VLATE CONTROL	The Envoy has two vlate mechanisms, selected by the vlate control bit.

CDSTAT	CD Status register	3F00h	Read only

Bit 0	FREADY	This bit is set after every frame received by the CD interface.
Bit 1	SYNCED1	This bit is set when the CD interface detects the first sync symbol after the DPLL has been enabled.
Bit 2	SYNCED2	This bit is set when the CD interface detects a sync coincidence (two sync symbols exactly 588) bits apart.
Bit 3	ONSYNC	This bit indicates that a valid sync symbol was received at the start of the previous frame. A window of +/- 6 bits is allowed for the system to re-synchronise after a multi frame error burst.
Bit 4	ONSYNC2	This is a more accurate version of the ONSYNC flag which could be cleared if the sync was early.
Bit 5	SLOW	This flag determines whether the DPLL has been running faster or slower than the obliterated data during a media defect.
Bit 6	SYNC	This bit is set if any sync symbol has been received since the last is flag determines whether the DPLL has been running faster or slower than the obliterated data during a media defect.

PLLFREQ	PLL initial frequency	3F02h	Write only

This eight bit register is used to specify the initial frequency the DPLL starts after it is enabled. The value is calculated as follows:

$$\text{Initial Frequency} = 256 * \text{Data Clock} / \text{CD clock}$$

The Data clock frequency is 4.32MHz for a 1X drive, 8.64MHz for a 2X drive etc. The CD clock frequency depends on the crystal chosen to clock the CD interface.

The initial frequency doesn't restrict the PLL range but the closer it is to the actual data the quicker it will lock. There is no reason why the PLL should not be enabled while the spindle motor is accelerating to its desired operating speed. Provided the head is focused and tracking the speed of the disk can be estimated from the MAX register. The initial value in these circumstances is given by:

$$\text{Initial Frequency} = 256 * 11 / \text{MAX}$$

MAX	CD DATA Maximum Pulse Width	3F02h	Read only

This register reflects the width of the maximum width data pulse. For CD data this would be an 11T pulse. The value read is the width of the widest pulse, in CD clock periods, since the last read. The register is reset to zero after being read. This register can be used to give a crude measure of disk speed prior to the DPLL being turned on and can be used to control the spindle motor until then.

Distortion and disk defects can give rise to long pulses so the value returned by the MAX register should be used with care. Once the DPLL is synchronised the input FIFO occupancy gives a better indication of disk speed.

DAC	Function	Output Pins	Address	Mode
DAC1	Focus Motor	foc	3F04h	Unipolar
DAC2	Tracking Motor	trk	3F06h	Unipolar
DAC3	Spindle Motor	spp,spm	3F08h	Bipolar
DAC4	Sled Motor	slp.slm	3F0Ah	Bipolar
DAC5	Laser Power	v8	3F0Eh	Unipolar
DAC6	Left DAC	v9	3F10h	Unipolar
DAC7	Right DAC	v10	3F12h	Unipolar
DAC8		v11	3F18h	Unipolar

ADC	Function	Input Pins	Address
ADC1	diode 0	di0	3F04h
ADC2	diode 1	di1	3F06h
ADC3	diode 2	di2	3F08h
ADC4	diode 3	di3	3F0Ah
ADC5		v4	3F0Ch
ADC6		v5	3F0Eh
ADC7		v6	3F0Eh
ADC8		v7	3F10h

Envoy has eight digital inputs which, combined with external 1 bit delta sigma converters, are used as analogue to digital converters. The external Sigma-delta converter uses an integrator, comparator, latch and one bit DAC to convert an analogue voltage into a stream of digital ones and zeros whose average value is proportional to the analogue voltage. The voltage is digitised by counting the ones in a given interval.

I2S1CNTRL I2S2CNTRL	I2S Control Register	3F0Ch 3F32h	Write Only

These registers controls the operation of the I2S interfaces. I2S is a serial interface for sending and receiving 16 bit stereo audio data and video data. It is mainly used for sending audio data to external DACs but its high bandwidth (up to around 10Mbit/sec) and low pin count (four pins) give it wider applicability.

The I2S interface can operate as either master or slave. As master the clock (I2SC) and word select (I2SW) pins become outputs and their timing is generated on chip. As slave these pins become inputs and the timing is determined externally. The data direction is independent of master and slave.

The data format, polarity, rate and master/slave mode are controlled by the I2SCNTRL register. Writing to this register resets the I2S interface.

Data is sent and received by means of registers I2STXL, I2STXR, I2SRXL and I2SRXR. Interrupt 6 (Vector 70h) is generated by the right to left transition of the word select signal for the first channel and Interrupt 7 (Vector 80h) is generated for the second channel. The processor should read both left and right received data and write both left and write transmit data at this time.

Bit 1	MASTER	When set this bit puts the chip into I2S master mode. The I2S clock and word select pins become outputs and the timing is generated internally.	
Bits 2-7	Rate	This six bit field determines the ratio between the I2S clock and the CD system clock. The I2S clock frequency is equal to:- CD system clock frequency / (rate + 1).	
Bit 8	Clock Polarity	This bit inverts the polarity of the I2S clock both as master and as slave.	
Bit 9	Word Select Polarity	This bit inverts the polarity of the I2S word select both as master and as slave.	
Bit 10	Data Polarity	This bit inverts the polarity of the I2S data on both transmit and receive.	
Bits 11-12	Data Format	These bits determine the data format as follows:	
		0	Philips
		1	Sony
		2	Matsushita

SYNCLR	Syndrome Clear	3F14h	Write Only

The Error Checking and Correction (ECC) mechanism relies on the efficient calculation of syndromes. To compute the syndrome of a number of bytes the syndrome is first cleared by writing (anything) to SYNCLR. The bytes are then "added" to the syndrome by executing the CHECK instruction on each byte in turn.

SYNLOW SYNHIGH	Syndrome Registers	3F14h 3F16h	Read Only

Once the syndrome of a number of bytes has been calculated it may be read at registers SYNLOW and SYNHIGH. The syndrome comprises of four bytes syn0, syn1, syn2 and syn3. When a byte is CHECK'd or "added" to the syndrome the previous syndrome bytes are first multiplied by successive powers of alpha. For instance after checking byte0, byte1 and byte2 the syndrome bytes would be:-

syn0 = byte0 * alpha^0 + byte1 * alpha^0 + byte2 * alpha^0
syn1 = byte0 * alpha^2 + byte1 * alpha^1 + byte2 * alpha^0
syn2 = byte0 * alpha^4 + byte1 * alpha^2 + byte2 * alpha^0
syn3 = byte0 * alpha^6 + byte1 * alpha^3 + byte2 * alpha^0

When the CD data is not corrupted, the syndromes will all be zero. Most errors in the data will result in one or more non zero syndromes which (using something like the above equation) can be used to identify the error and correct the data.

Syn0 is the low byte of SYNLOW. Syn1 is the high byte of SYNLOW. Syn2 is the low byte of SYNHIGH and Syn3 is the high byte of SYNHIGH.

CDERROR	CD Error Register	3F18h	Write Only

This register contains left and right error bits that are multiplexed onto the ErrorFlag output pin. This allows the software to flag errors to external hardware.

Bit	0	1
	Left Channel Error	Right Channel Error

SUBCODE	Subcode Register	3F18h	Read Only

There are eight subcode channels. One bit of each channel is carried in a CD frame so the data rate is 7.35kbps per channel. Bits 0 to 7 of the subcode register correspond to the data in subcode channels as follows:

Bit	0	1	2	3	4	5	6	7
Channel	P	Q	R	S	T	U	V	W

Bit eight of SUBCODE is set when the CD frame carries the first of the two subcode sync symbols. Subcodes form 98 bit packets which are distributed across 98 CD frames. This bit is set for the first frame of 98.

The subcode register is updated once per frame so it could be read when a frame has been DMA'd into memory (and interrupted the processor) or when the FREADY bit in CDSTAT is set.

All further analysis and use of the subcodes must be supported in software.

CDTIMER	CD Sample rate timer	3F1Ah	Write Only

The CD Sample rate timer controls the sample rate of the DACs and ADCs. The sample rate period in CD system clock cycles is given by:-

$$4096 - N$$

where N is the value written to CDTIMER.

The timer is designed to support a sample rate around the CD-Audio sample frequency of 44.1kHz. Interrupt 0 (vector 10h) is generated once per sample. A sample rate pulse may be output on pin v34 by setting bit EXSMP in register DCNTRL.

PLL	Phase Locked Loop register	3F1Ah	Read Only

When the Digital Phase Locked Loop (DPLL) is locked to the CD data this register gives the instantaneous data clock frequency which depends on the instantaneous spindle speed. The value in this register is given by:

$$65536 * \text{Data Clock} / \text{CD Clock}$$

The Data Clock frequency should be 4.32MHz and the CD Clock frequency depends on the crystal used for the CD interface logic. The difference from the ideal value can be used as the proportional error term in the spindle servo.

I2S1TXL	I2S Transmit Data	3F1Ch	Write Only
I2S1TXR		3F1Eh	
I2S2TXL		3F2Eh	
I2S2TXR		3F30h	

These registers hold the left and right I2S transmit data for the 2 I2S channels. They should be loaded after the right to left transition of the I2S word select (which generates interrupt 6). The data is transmitted following the next right to left transition of word select. The left data is sent first.

I2S1RXL	I2S Receive Data	3F1Ch	Read Only
I2S1RXR		3F1Eh	
I2S2RXL		3F2Eh	
I2S2RXR		3F30h	

These registers hold the left and right I2S received data for the 2 I2S channels. They should be read after the right to left transition of the I2S word select (which generates interrupt 6).

OUT0 OUT1	General Purpose Outputs	3F20h 3F22h	Write Only

EN0 EN1	General Purpose Output Enables	3F26h 3F28h	Write Only

IN0 IN1	General Purpose Inputs	3F20h 3F22h	Read Only
OUTSEL0 OUTSEL1	General Purpose Output Selects	3F24h 3F2Ah	Write Only

Fig. 13 shows the digital phased lock loop (PLL) circuit used to recover the clock. The overall gain of the PLL can be changed by adjusting the scaling factors or coefficients. This allows the PLL to acquire lock quickly but ignore noise once locked.

The clock recovery PLL 20 includes a phase error latch 70, a proportional constant multiplier 72, an integral constant multiplier 74, an integrator 76 that contains an adder 78 and a latch 80, a proportional and integral sum adder 82 and a recovered clock generator 84 that contains an adder 86 and a latch 88. The error latch 70 has a data input 90 for CD data and a recovered clock input 92, which is connected to the output of the recovered clock generator 84 to form a feedback loop. The phase error latch 70 and the latches 80, 88 in the integrator 76 and the recovered clock generator 84 are connected to the system clock 94.

CD data is recorded on the CD with transitions representing a logical '1'. When there is a transition in the data stream, the error latch 70 is loaded with a value representing the time difference between the expected arrival time of the transition, based on the recovered clock frequency, and the actual arrival time. An error signal is thus generated at an output 96, which is used to correct the recovered clock frequency, to allow the CD clock to be generated from the transitions on the CD.

The proportional constant multiplier 72 multiplies the error signal from the error latch 70 by a constant, which may be varied under software control. This constant, together with the integral constant, controls noise rejection and clock acquisition of the loop. The integral constant multiplier 74 operates in a similar way to the proportional constant multiplier 72, multiplying the error signal from the error latch 70 by another constant value.

The output of the integral constant multiplier 74 is connected to the integrator 76. The adder 78 adds the current value of the integrator latch 80 to the error signal output of the integral constant multiplier 74, thus acting as an integrator. When a data transition is detected on the CD, the integrator latch 80 is updated.

The proportional and integral sum adder 82 adds the proportional and integral terms together to generate the control signal for the recovered clock generator 84. The value of this control signal will vary with the timing error between the expected clock and the actual clock position, allowing the recovered clock to be locked to the CD.

The recovered clock generator 84 generates both the recovered CD clock and the error signal for the error latch 70. The adder 86 adds the output from the proportional and integral sum adder 82 to the value in the latch 88. It therefore acts as a counter counting up in increments of the value from the proportional and integral sum adder 82. The recovered clock is generated from the most significant bit of the counter.

The PLL circuit thus changes the recovered clock frequency until the positive transition on the recovered clock coincides with transitions in the data from the CD.

For the integral term the error is divided by a power of two between 2^7 and 2^18. Bits 0-7 control the integral term. Bits 4-7 select a gain during capture (fast) and bits 0-3 select a gain during hold (slow). The bit fields select the gain according to the following table.

Bits	Gain
0	2^18
1	2^17
2	2^16
3	2^15
4	2^14
5	2^13
6	2^12
7	2^11
8	2^10
9	2^9
10	2^8
11	2^7

Bits 8-15 control the proportional term. Bits 12-15 select a gain during capture (fast) and bits 8-11 select a gain during hold (slow). The bit fields select the gain according to the following table.

Bits	Gain
0	2^15
1	2^14

Bits	Gain
2	2^13
3	2^12
4	2^11
5	2^10
6	2^9
7	2^8
8	2^7
9	2^6
10	2^5
11	2^4

IRT	Infra Red Timer	3F34h	Write

This register holds the infra-red timer. The infra red input is sampled at a rate equal to the processor clock, divided by the 12 bit value held in this register (-1 ??). The infra red input is filtered, and is only recognized to have changed state when three consecutive samples are the same. When the input changes state, the NVP is interrupted on level 3.

INK	Link Registers	3F40h	Read/Write

These 16 bit registers allows rapid bi-directional communication between the test interface and NVP when the Envoy has TMODE enabled. Data written by NVP may be read by the test interface and data written by the test interface may be read by NVP. Transfers from NVP to the test interface may be concurrent with transfers from the test interface to NVP.

SMCADDR	Program Address Register	3F44h	Write Only

This register is part of a mechanism for allowing NVP to write into its own program memory. The program memory is separate to the data memory and would otherwise be saturated by instruction fetches. The SMC (self modifying code) instruction creates gaps in the instruction flow and allows the location specified by this register to be written with the data in the program data registers.

The program address register is shared with the host. The host should therefore not write to the address register while NVP is running. Because the register is also used to address the bootstrap ROM it is actually a byte address. To modify instruction N in program memory the value 4N must be written into the program address register.

SMCDLOW SMCDHI	Program Data Registers	3F46h 3F48h	Write Only

These registers provide the data (new instruction) for the SMC (self modifying code) instruction described under the Program Address Register above. SMCDLOW holds the low 16 bits of the instruction to be written. SMCDHI holds the high 16 bits.

DCNTRL	DMA Control Register	3F50h	Write Only

Bit 9	EXDMA	When set this bit makes NVP's first DMA channel (the byte channel) available to an external data source. The DMA request input is pin v24 and a lookahead DMA acknowledge is output on pin v22. A DMA transfer is requested by setting v24 high for one clock cycle. Byte wide data must be enabled onto both halves of the data bus in the cycle(s) following the assertion of signal v22. The data will be written into the memory location specified by the DMA page and address registers.

5.4 NVP INSTRUCTION SET OVERVIEW

Each instruction is 32 bits and comprises of a number of fixed fields as follows.

Bits 0-15	Operand	This 16 bit value may be used as immediate data, direct address or offset.	
Bits 16-20	Op code	This 5 bit field determines which of 32 operations will be performed. Most opcodes use the other fields, size, source & destination in a regular way. Some borrow these fields for specific purposes.	
Bit 21	Mode	This bit is used to determine whether loads and stores should use direct (nn) or indirect (Rx +nn) addressing. It is also used to determine whether ALU operations should use immediate (nn) or register (Rx) source operands.	
Bits 22-23	Size	This bit determines the size of the operand as follows	
		0	byte
		1	word
		2	long
		For loads and stores bit 23 is used to force modular addressing as follows	
		0	byte
		1	word
		2	byte modular
		3	word modular

Bits 24-27	Source	This specifies a source register for ALU operations or an address register for loads and stores. The source register also specifies the shift count in variable shift instructions. The source field is used to further decode instructions sharing opcode zero.		
Bits 28-31	Destination	This specifies a destination register for ALU operations or a destination/source register for loads/stores. The destination field is also used to encode a condition code for JUMP and CALL as follows.		
		0	Zero	equal
		1	Non zero	not equal
		2	Carry	B > A
		3	NC	A >= B
		4	C or Z	B > = B
		5	NC & NZ	A > B
		6	Always	
		7	Never	

5.5 INSTRUCTION LISTING

NOP	No Operation
EI	Enable Interrupts
DI	Disable Interrupts
RET	Return from Subroutine
MOV BMA,R12	Set byte channel DMA address
MOV R12,WMC	Read word channel word count
MOV SP,R12	Set Stack Pointer
MOV R12,SP	Read Stack Pointer
RETI	Return from Interrupt
MOV BMC,R12	Set byte channel byte count
MOV R12,BMC	Read byte channel byte count.
MOV BMP,R12	Set byte channel page
MOV MASK,R12	Set interrupt mask
MOV TIME,R12	Set Timer
WAITM	Wait for Memory
MOV PAGE,R12	Set page register
RESPOLY	Reset Scrambler
SCRAMBLE	Scramble R12
MOV R12,TIME	Read Timer
RESEDC	Reset EDC
MOV EDC,R12	Write to EDC
READEDC	Read EDC

SMC	Self Modifying Code
MOV WMA,R12	Set word channel DMA address
MOV WMC,R12	Set word channel word count
MOV WMP,R12	Set word channel page
LOAD.B d,(nn)	Load byte direct.
LOAD.W d,(nn)	Load word direct.
MLOAD.B d,(nn)	Load byte modular direct.
MLOAD.W d,(nn)	Load word modular direct.
LOAD.B d,(s+nn)	Load byte indirect.
LOAD.W d,(s+nn)	Load word indirect.
MLOAD.B d,(s+nn)	Load byte modular indirect.
MLOAD.W d,(s+nn)	Load word modular indirect.
STORE.B (nn),d	Store byte direct.
STORE.W (nn),d	Store word direct.
MSTORE.B (nn),d	Store byte modular direct.
MSTORE.W (nn),d	Store word modular direct.
STORE.B (s+nn),d	Store byte indirect.
STORE.W (s+nn),d	Store word indirect.
MSTORE.B (s+nn),d	Store byte modular indirect.
MSTORE.W (s+nn),d	Store word modular indirect.
XLAT.B d,(s+nn)	Translate byte
JUMP cc,nn	Jump direct
JUMP cc,s	Jump indirect
CALL cc,nn	Call direct
CALL cc,s	Call indirect
CHECK.B d,(nn)	Load & check byte direct
MCHECK.B d,(nn)	Load & check byte modular direct
CHECK.B d,(s+nn)	Load & check byte indirect
MCHECK.B d,(s+nn)	Load & check byte modular indirect
ADD.B d,nn	Add byte Immediate
ADD.W d,nn	Add word Immediate
ADD.B d,s	Add byte Register
ADD.W d,s	Add word Register
ADD.W d,s	Add long Register
SUB.B d,nn	Subtract byte Immediate
SUB.W d,nn	Subtract word Immediate
SUB.B d,s	Subtract byte Register
SUB.W d,s	Subtract word Register
SUB.W d,s	Subtract long Register
ADC.B d,nn	Add with Carry byte Immediate

ADC.W d,nn	Add with Carry word Immediate
ADC.B d,s	Add with Carry byte Register
ADC.W d,s	Add with Carry word Register
ADC.W d,s	Add with Carry long Register
SBC.B d,nn	Subtract with Carry byte Immediate
SBC.W d,nn	Subtract with Carry word Immediate
SBC.B d,s	Subtract with Carry byte Register
SBC.W d,s	Subtract with Carry word Register
SBC.W d,s	Subtract with Carry long Register
AND.B d,nn	And byte Immediate
AND.W d,nn	And word Immediate
AND.B d,s	And byte Register
AND.W d,s	And word Register
AND.W d,s	And long Register
CMP.B d,nn	Compare byte Immediate
CMP.W d,nn	Compare word Immediate
CMP.B d,s	Compare byte Register
CMP.W d,s	Compare word Register
CMP.W d,s	Compare long Register
OR.B d,nn	Or byte Immediate
OR.W d,nn	Or word Immediate
OR.B d,s	Or byte Register
OR.W d,s	Or word Register
OR.W d,s	Or long Register
XOR.B d,nn	Exclusive Or byte Immediate
XOR.W d,nn	Exclusive Or word Immediate
XOR.B d,s	Exclusive Or byte Register
XOR.W d,s	Exclusive Or word Register
XOR.W d,s	Exclusive Or long Register
ADDM.B d,nn	Add byte modulo 255 Immediate
ADDM.B d,s	Add byte modulo 255 Register
ADDM.B d,nn	Add byte modulo 255 Immediate
ADDM.B d,s	Add byte modulo 255 Register
SUBM.B d,nn	Subtract byte modulo 255 Immediate
SUBM.B d,s	Subtract byte modulo 255 Register
SUBM.B d,nn	Subtract byte modulo 255 Immediate
SUBM.B d,s	Subtract byte modulo 255 Register
SLZ.B d,nn	Shift left zero byte immediate
SLC.B d,nn	Shift left carry byte immediate
SLM.B d,nn	Shift left msb byte immediate

SLL.B d,nn	Shift left lsb byte immediate
SRZ.B d,nn	Shift right zero byte immediate
SRC.B d,nn	Shift right carry byte immediate
SRM.B d,nn	Shift right msb byte immediate
SRL.B d,nn	Shift right lsb byte immediate
SLZ.B d,s	Shift left zero byte register
SLC.B d,s	Shift left carry byte register
SLM.B d,s	Shift left msb byte register
SLL.B d,s	Shift left lsb byte register
SRZ.B d,s	Shift right zero byte register
SRC.B d,s	Shift right carry byte register
SRM.B d,s	Shift right msb byte register
SRL.B d,s	Shift right lsb byte register
SLZ.W d,nn	Shift left zero word immediate
SLC.W d,nn	Shift left carry word immediate
SLM.W d,nn	Shift left msb word immediate
SLL.W d,nn	Shift left lsb word immediate
SRZ.W d,nn	Shift right zero word immediate
SRC.W d,nn	Shift right carry word immediate
SRM.W d,nn	Shift right msb word immediate
SRL.W d,nn	Shift right lsb word immediate
SLZ.W d,s	Shift left zero word register
SLC.W d,s	Shift left carry word register
SLM.W d,s	Shift left msb word register
SLL.W d,s	Shift left lsb word register
SRZ.W d,s	Shift right zero word register
SRC.W d,s	Shift right carry word register
SRM.W d,s	Shift right msb word register
SRL.W d,s	Shift right lsb word register
SLZ.L d,nn	Shift left zero long immediate
SLC.L d,nn	Shift left carry long immediate
SLM.L d,nn	Shift left msb long immediate
SLL.L d,nn	Shift left lsb long immediate
SRZ.L d,nn	Shift right zero long immediate
SRC.L d,nn	Shift right carry long immediate
SRM.L d,nn	Shift right msb long immediate
SRL.L d,nn	Shift right lsb long immediate
SLZ.L d,s	Shift left zero long register
SLC.L d,s	Shift left carry long register
SLM.L d,s	Shift left msb long register

SLL.L d,s	Shift left lsb long register
SRZ.L d,s	Shift right zero long register
SRC.L d,s	Shift right carry long register
SRM.L d,s	Shift right msb long register
SRL.L d,s	Shift right lsb long register
MOV.B d,nn	Move byte immediate
MOV.B d,s	Move byte register
MOV.W d,nn	Move word immediate
MOV.W d,s	Move word register
MOV.L d,s	Move long register
IMACC.W d,r0,r14	Initial Multiply Accumulate word
IMACC.L d,r0,r14	Initial Multiply Accumulate long
MACC.W d,r0,r14	Multiply Accumulate word
MACC.L d,r0,r14	Multiply Accumulate long
ELOAD.B d,(s)	Load byte Extended
ELOAD.W d,(s)	Load word Extended
ESTORE.B (s),d	Store byte Extended
ESTORE.W (s),d	Store word Extended
GMUL d,nn	Galois Multiply
GMUL d,s	Galois Multiply

INSTRUCTION DETAILS

In the descriptions of instructions which follow the flags are unchanged unless explicitly mentioned.

NOP		No operation			
Destination 0	Source 0	Size X	Mode X	Opcode 0	Operand X

This instruction has no effect on the registers or flags. It could be used as the instruction after a JUMP or CALL if nothing useful can be done.

EI		Enable Interrupt			
Destination 0	Source 1	Size X	Mode X	Opcode 0	Operand X

This instruction enables interrupts. The following instruction is always executed before an interrupt may occur.

DI		Disable Interrupt			
Destination 0	Source 2	Size X	Mode X	Opcode 0	Operand X

This instruction disables interrupts. This instruction takes immediate effect. The following instructions will not be interrupted.

RET		Return from Subroutine			
Destination 0	Source 3	Size X	Mode X	Opcode 0	Operand X

This instruction decrements the stack pointer by two then loads the PC but not the flags from the stack. Because the flags are not restored by RET they can be used as the result of a subroutine. Pipelining causes the two instructions following a RET to be executed before execution returns. The programmer should therefore execute the RET two instructions early. Interrupts are temporarily disabled during the two instructions following a RET. This means that they will always be executed together. It is possible for the processor to return and immediately re-stack the return address of the underlying code because of a pending interrupt.

MOV BMA,R12		Set byte channel DMA address			
Destination 0	Source 4	Size X	Mode X	Opcode 0	Operand X

This instruction loads the first DMA channel (the byte channel) address register with the contents of R12. The address register is an unbuffered counter and must be re-loaded in software at the start of every block to be DMA'd.

MOV R12,WMC		Read word channel word count			
Destination 0	Source 5	Size X	Mode X	Opcode 0	Operand X

This instruction loads R12 with the word channel word count. The Word count is an unbuffered 15 bit counter which counts from the loaded value to zero then stops. This instruction has been provided to allow the processor to monitor the progress of the word channel DMA. Interrupt 7 (vector 80h) is generated at the end of the block transfer.

DMA operations may not cross a 64k boundary so this counter is restricted to 15 bits (32k words).

MOV SP,R12		Set Stack Pointer			
Destination 0	Source 6	Size X	Mode X	Opcode 0	Operand X

This instruction loads the stack pointer from R12. The stack pointer is a 16 bit register which addresses a location in the first 64k of data memory. The stack is used to store return addresses and flags for interrupt service routines and subroutines. It is not used for PUSHing and POPing registers. The stack builds from a low address to a high address as more addresses are PUSH'd onto the stack. The stack pointer points to the next unused location on the stack so uses post-increment and pre-decrement.

Bit zero of the stack pointer is always zero and writing non zero will have no effect. Only bits 1 to 6 can count which provides a stack size of 64 entries. The stack will wrap around if too many entries are PUSH'd.

MOV R12,SP		Read Stack Pointer			
Destination 0	Source 7	Size X	Mode X	Opcode 0	Operand X

This instruction loads R12 with the stack pointer.

RETI		Return from Interrupt			
Destination 0	Source 8	Size X	Mode X	Opcode 0	Operand X

This instruction decrements the stack pointer by two then loads the PC and flags from the stack. Interrupts are enabled by RETI but as with RET interrupts are temporarily disabled during the two instructions following the RETI. Although it is possible for the processor to return and immediately re-stack the return address of the underlying code (because of a pending interrupt) this mechanism will prevent stack crawl.

As with RET pipelining causes the two instructions following a RETI to be executed before execution returns. The programmer should therefore execute the RETI two instructions early. The two instructions after the RETI should not modify the flags.

MOV BMC,R12		Set byte channel byte count			
Destination 0	Source 9	Size X	Mode X	Opcode 0	Operand X

This instruction sets the byte channel byte count from register R12. The byte count is an unbuffered 16 bit up counter which must be re-loaded at the start of every DMA block. It should be loaded with the value 65536 - block size.

DMA may not cross a 64k boundary so the count is restricted to 64k.

MOV R12,BMC		Read byte channel byte count			
Destination 0	Source 0Ah	Size X	Mode X	Opcode 0	Operand X

This instruction loads R12 with the byte channel byte count. The byte count is an unbuffered 16 bit counter which counts from the loaded value to zero then stops. This instruction has been provided to allow the processor to monitor the progress of the byte channel DMA. Interrupt 2 (vector 30h) is generated at the end of the block transfer.

MOV BMP,R12		Set byte channel page			
Destination 0	Source 0Bh	Size X	Mode X	Opcode 0	Operand X

This instruction loads the byte channel page register from R12. The byte channel page register extends the byte channel address from 16 bits to 32 bits. This allows the byte channel to transfer data into any 64k block in memory.

MOV MASK,R12		Set interrupt mask			
Destination 0	Source 0Ch	Size X	Mode X	Opcode 0	Operand X

This instruction loads the interrupt mask register from R12. Each bit of the interrupt mask register corresponds to an interrupt source. If the bit is set then the corresponding interrupt is enabled. The mask works at the input stage of the interrupt mechanism. If there is a pending interrupt from a given interrupt source then clearing the corresponding bit in the mask register will not clear the pending interrupt. The pending interrupt can only be cleared by vectoring to the corresponding interrupt service routine.

If a bit in the interrupt mask register is set then any interrupts which occurred while the bit was clear will be ignored.

MOV TIME,R12		Set Timer			
Destination 0	Source 0Dh	Size X	Mode X	Opcode 0	Operand X

This instruction loads the timer from R12. The timer is an unbuffered 16 bit counter which increments every clock cycle. The timer generates interrupt 5 (vector 60k) when it overflows from 0FFFFh to 0. This instruction allows it to be set which might be used to generate a pulse of a well defined length or to generate an interrupt at a frequency other than the CD sample rate.

WAITM		Wait for Memory			
Destination 0	Source 0Eh	Size X	Mode X	Opcode 0	Operand X

This instruction suspends execution until any external memory transfer has completed. External memory comes in three types: quick, slow and DRAM. Quick memory has a one clock cycle time. Slow memory has a cycle time between two and nine clock cycles

depending on the configuration bits set during reset. DRAM has four cycle times depending on whether the transfer is within the same row (page mode cycle) and on whether the DRAM is fast or slow. The following table gives DRAM cycle times.

	New ROW	Same ROW
Slow	8	3
Fast	6	2

WAITM is useful for suspending execution until a register has been loaded from external memory. The processor can overlap internal memory transfers with external transfers so it is possible to keep the processor busy while performing slow external transfers.

There is an implied WAITM associated with all load and store operations to external memory. This means that if a block of data must be transferred to external memory then WAITM is not necessary between loads and stores.

MOV PAGE,R12		Set page register			
Destination 0	Source 0Fh	Size X	Mode X	Opcode 0	Operand X

This instruction loads the page register from register R12. The page register is a sixteen bit register which is used to extend 16 bit addresses to 30 bits. 16 bit addresses are either direct addresses from the instruction operand field or indirect addresses which are the sum of a register plus the instruction operand.

If the address is in the range 0-3FFFh then the address is extended with zeros to address the first 16k of memory. This includes the fast internal memory and all the internal memory mapped I/O.

If the address is in the range 4000h-FFFFh then the top two bits are added to the page register to form the top 16 bits of a 30 bit address. Here are some examples.

Page	16 bit	30 bit	
1234h	2345h	00002345h	First 16k
1234h	4567h	048D4567h	
003Fh	4000h	00100000h	start of quick RAM
007Fh	4001h	00200001h	start of slow RAM
00FFh	4001h	00400001h	start of DRAM
00FFh	8000h	00404000h	DRAM + 4000h
03FCh	FFFFh	00FFFFFFh	end of DRAM

The page mechanism effectively provides two windows into memory. One is a fixed 16k window into the fast internal memory and I/O space. The other is a moveable 48k window which can be put on any 16k boundary in memory.

RESPOLY		Reset Scrambler			
Destination 1	Source 0	Size X	Mode X	Opcode 0	Operand X

This instruction resets the polynomial used to unscramble the data in CD-ROM data blocks. This instruction must be executed before unscrambling the first byte of CD-ROM data.

SCRAMBLE		Scramble R12			
Destination 1	Source 1	Size X	Mode X	Opcode 0	Operand X

This instruction un-scrambles the byte in R12. All CD-ROM data excluding the sync bytes is scrambled with the scrambler mechanism shown in Fig. 14.

The data scrambler uses a 15 bit shift register 100, which shifts data right (i.e. towards the least significant bit) every clock cycle. The most significant bit is generated by an adder 102, which adds the two least significant bits of the shift register 100. This feedback generates a pseudo-random code from the least significant bit of the shift register, which has the maximum spacing between codes being repeated.

The least significant bit of the shift register is added to the data stream by a second adder 104 to spread the spectrum of the data stream and make it less susceptible to errors on the CD.

The value in the shift register 100 is preset with a fixed value when initialising the circuit to ensure it generates the correct code.

The scramble instruction un-scrambles the data on a byte by byte basis.

MOV R12,TIME		Read Timer			
Destination 1	Source 2	Size X	Mode X	Opcode 0	Operand X

This instruction loads R12 from the timer. The timer is an unbuffered 16 bit counter which increments every clock cycle. The timer generates interrupt 5 (vector 60k) when it overflows from 0FFFFh to 0. This instruction allows it to be read which might be used to measure an interval of time with greater precision than would be possible using, say, the sample rate interrupt.

RESEDC		Reset EDC			
Destination 1	Source 3	Size X	Mode X	Opcode 0	Operand X

This instruction resets the mechanism used to detect errors in CD-ROM data. This 32 bit polynomial is different to the scrambling polynomial and provides a final check that data

corrected by the underlying ECC mechanism is OK. This instruction must be executed before computing the EDC polynomial.

MOV EDC,R12		Write to EDC			
Destination 1	Source 4	Size X	Mode X	Opcode 0	Operand X

This instruction feeds the byte in R12 through the EDC mechanism generator. When all the data has been fed through a non zero value (as read by READEDC) indicates that the data contains an error.

The polynomial used is:

$$P(x) = (x^16 + x^15 + x^2 + 1) * (X^16 + X^2 + X + 1)$$

The EDC logic is equivalent to a 32 bit shift register which shifts data bits in from the msb to the lsb. If the bit shifted out is set then the shift register contents is xor'd with the longword D8018001h.

READEDC		Read EDC			
Destination 1	Source 5	Size X	Mode X	Opcode 0	Operand X

This instruction sets the zero flag if the EDC mechanism contains a non zero value. This indicates that the data was erroneous.

SMC		Self Modifying Code			
Destination 1	Source 6	Size X	Mode X	Opcode 0	Operand X

This instruction suspends program execution for one clock cycle while a location in program memory is modified. The location is identified by the value written to I/O register SMCADDR (which should be four times the index of that instruction). The new instruction is the value written to I/O registers SMCDLOW and SMCDHI.

MOV WMA,R12		Set word channel DMA address			
Destination 1	Source 7	Size X	Mode X	Opcode 0	Operand X

This instruction loads the word channel address register with the contents of R12. The address register is an unbuffered counter and must be re-loaded in software at the start of every block to be DMA'd.

MOV WMC,R12		Set word channel word count			
Destination 1	Source 8	Size X	Mode X	Opcode 0	Operand X

This instruction sets the word channel word count from register R12. The word count is an unbuffered 16 bit up counter which must be re-loaded at the start of every DMA block. It should be loaded with the value 32768 - block size.

DMA may not cross a 64k boundary so the count is restricted to 32k.

MOV WMP,R12		Set word channel page			
Destination 1	Source 9	Size X	Mode X	Opcode 0	Operand X

This instruction loads the word channel page register from R12. The byte channel page register extends the byte channel address from 16 bits to 32 bits. This allows the word channel to transfer data into any 64k block in memory.

LOAD d,(nn) LOAD d,(s+nn)		Load Direct Load Indirect			
Destination d	Source s	Size 0 byte 1 word 2 byte mod. 3 word mod	Mode 0 (direct) 1 (indirect)	Opcode 1	Operand nn

This instruction loads the destination register from memory. The load may be either a byte or word depending on the size field. (The size field is also used to specify modular addressing).

If the mode bit is set (indirect mode) a 16 bit address is formed from the sum of the source register and the instruction operand. If the mode bit is clear (direct mode) the operand is used directly as a 16 bit address. The 16 bit address is combined with the page register as described above to form a 30 bit address.

Because of pipelining the memory read starts during the cycle after the instruction executes. Addresses in the first 1Mbyte refer to internal memory and internal I/O which have a one clock cycle time. Data loaded from internal memory can therefore be used one instruction after the LOAD. In fact the previous value of the destination register is available in the instruction after the LOAD. This helps by allowing LOADs to be advanced so that memory based data is available when needed.

```
LOAD.B    r4,(new_value)
STORE.B   (old_value),r4
ADD.B     r6.r4
```

The above sequence of instructions stores the initial value of r4 at (old_value) and the value at (new_value) is added to r6.

If the address is above 1M byte then the memory is external. If the address is above 2M byte the memory is external and the transfer time is longer than one clock and the mechanisms described under the WAITM instruction should be used to ensure that the data has loaded.

It is not possible to load longwords so two combinations of the size field are used to specify modular addressing as follows.

size field	operation
0	load byte
1	load word
2	load byte modular
3	load word modular

Modular addressing is used to contain a pointer within a buffer. It works by comparing the 16 bit address formed above with the 16 bit address in register R2. If the address is greater than or equal to R2 then the value in R0 is subtracted from the address. R0 should contain the length of the buffer and R2 the address of the byte immediately after the buffer.

STORE (nn),d STORE (s+nn),d		Store Direct Store Indirect			
Destination d	Source s	Size 0 byte 1 word 2 byte mod. 3 word mod	Mode 0 (direct) 1 (indirect)	Opcode 2	Operand nn

This instruction stores the destination register in memory. The operand size and address calculation are all as for the LOAD instruction.

The memory cycle time is not as significant to the use of the STORE instruction because the address and data are latched by the memory controller. This allows the destination and source registers to be used immediately even though the STORE may take several cycles to complete.

XLAT.B d,(s+nn)		Translate			
Destination d	Source s	Size 0 Byte	Mode 1 indirect	Opcode 3	Operand nn

This instruction is a variation of LOAD byte indirect. Instead of forming the address from a word register it is formed from a byte register. This is particularly useful for looking up powers and logs in error correction.

So for instance

```
XLAT.B    r5,(r4+log)
```

loads r5 with the log of r4.

The same delays which apply to LOAD also apply to XLAT.

JUMP cc,nn JUMP cc,s		Jump direct Jump indirect			
Destination cc	Source s	Size Word	Mode 1 direct 0 indirect	Opcode 4	Operand nn

This instruction reloads the PC from either the operand (direct jump) or the source register (indirect jump) if the condition encoded in the destination field is met by the flags. Because of pipelining the instruction after the jump is always executed whether the jump is taken or not.

Interrupts are temporarily disabled during the instruction after a jump.

The state of the flags which enable the jump are encoded in the destination field as follows.

Destination	Flags	Meaning
0	Zero	equal
1	Non zero	not equal
2	Carry	B > A
3	NC	A >= B
4	C or Z	B >= A
5	NC & NZ	A > B
6	Always	
7	Never	

CALL cc,nn CALL cc,s		Call direct Call indirect			
Destination cc	Source s	Size Word	Mode 1 direct 0 indirect	Opcode 5	Operand nn

This instruction reloads the PC from either the operand (direct jump) or the source register (indirect jump) if the condition encoded in the destination field is met by the flags. The old PC and the flags are written to the memory location identified by the stack pointer and the stack pointer is incremented by two.

As with JUMP the instruction after a CALL is always executed whether the call is taken or not and interrupts are temporarily disabled for the instruction following the CALL. The Condition code is encoded as for JUMP.

CALL causes a 16 bit write to data memory. Bits 0-13 hold the program counter for the next instruction. Bit 14 holds the carry flag and bit 15 holds the zero flag.

CHECK.B d,(nn) CHECK.B d,(s+nn)		Load & Add to Syndrome Direct Load & Add to Syndrome Indirect			
Destination d	Source s	Size Byte Byte mod	Mode 0 direct 1 indirect	Opcode 6	Operand nn

This instruction is identical to LOAD byte except that the byte read from memory is also added to the ECC syndrome. This operation is described more fully in the description of the SYNLOW and SYNHIGH registers.

The destination register of the CHECK instructions should be an even register.

CHECK is particularly useful for computing the syndromes at C1, C2, P and Q stages of error correction. In the following sequence of instructions it is combined with modular addressing in a way which is limited only by memory bandwidth so is as fast as a dedicated hardware solution.

```
mcheck.b    r8,(r4+56+del_buf)
mcheck.b    r10,(r4+56+57+del_buf)
mcheck.b    r12,(r4+56+114+del_buf)
mcheck.b    r14,(r4+56+171+del_buf)
```

ADD d,nn ADD d,s		Add Immediate Add Register			
Destination d	Source s	Size 0 Byte 1 Word 2 Long	Mode 0 Register 1 Immediate	Opcode 8	Operand nn

This instruction adds the immediate data or source register to the destination register and sets the flags appropriately.

If the mode bit is set (immediate) then the operand is added to the destination register. If the mode bit is clear (register) then the source register is added to the destination register.

The ADD instruction can add bytes, words or longs depending on the size field in the instruction. It can only add longwords in register mode as the immediate data is only 16 bits.

The zero flag is set if the result is zero whether it be a byte, word or longword. The carry flag is set if there is a carry out as a result of the addition. That is if the sum is greater than 255 for byte operands, 65,535 for word operands or 4,294,967,295 for long operands.

SUB d,nn SUB d,s		Sub Immediate Sub Register			
Destination d	Source s	Size 0 Byte 1 Word 2 Long	Mode 0 Register 1 Immediate	Opcode 9	Operand nn

This instruction subtracts the immediate data or source register from the destination register and sets the flags appropriately.

If the mode bit is set (immediate) then the operand is subtracted from the destination register. If the mode bit is clear (register) then the source register is subtracted from the destination register.

The SUB instruction can subtract bytes, words or longs depending on the size field in the instruction. It can only subtract longwords in register mode as the immediate data is only 16 bits.

The zero flag is set if the result is zero whether it is a byte, word or longword. The carry flag is set if there is a borrow as a result of the subtraction. That is if the result is less than zero.

ADC d,nn ADC d,s		Add with Carry Immediate Add with Carry Register			
Destination d	Source s	Size 0 Byte 1 Word 2 Long	Mode 0 Register 1 Immediate	Opcode 0Ah	Operand nn

This instruction is identical to the ADD instruction but the carry flag is also added in at the least significant bit. If the carry bit is clear the result is the same as for ADD. If the carry bit is set the result is one greater than for ADD.

SBC d,nn SBC d,s		Subtract with Carry Immediate Subtract with Carry Register			
Destination d	Source s	Size 0 Byte 1 Word 2 Long	Mode 0 Register 1 Immediate	Opcode 0Bh	Operand nn

This instruction is identical to the SUB instruction but the carry flag (borrow) is also subtracted from the least significant bit. If the carry bit is clear the result is the same as for SUB. If the carry bit is set the result is one less than for SUB.

AND d,nn AND d,s		Bitwise AND Immediate Bitwise AND Register			
Destination d	Source s	Size 0 Byte 1 Word	Mode 0 Register 1 Immediate	Opcode 0Ch	Operand nn

This instruction replaces the destination register with the logical AND of the destination register and the immediate operand or the source register.

If the mode bit is set (immediate) then the immediate data is used. If the mode bit is clear (register) then the source register is used.

The operands may be either byte or word. The zero flag is set if the result is zero.

CMP d,nn CMP d,s		Compare Immediate Compare Register			
Destination d	Source s	Size 0 Byte 1 Word 2 Long	Mode 0 Register 1 Immediate	Opcode 0Dh	Operand nn

This instruction compares the immediate data or source register with the destination register and sets the flags appropriately. It is identical to the SUB instruction but the destination register is not modified.

If the mode bit is set (immediate) then the operand is compared with the destination register. If the mode bit is clear (register) then the source register is compared with the destination register.

The CMP instruction can compare bytes, words or longs depending on the size field in the instruction. It can only compare longwords in register mode as the immediate data is only 16 bits.

The zero flag is set if the result would be zero whether it is a byte, word or longword. The carry flag is set if there is a borrow as a result of the compare. That is if the result would be less than zero.

OR d,nn OR d,s		Bitwise OR Immediate Bitwise OR Register			
Destination d	Source s	Size 0 Byte 1 Word	Mode 0 Register 1 Immediate	Opcode 0Eh	Operand nn

This instruction replaces the destination register with the logical OR of the destination register and the immediate operand or the source register.

If the mode bit is set (immediate) then the immediate data is used. If the mode bit is clear (register) then the source register is used.

The operands may be either byte or word. The zero flag is set if the result is zero.

XOR d,nn XOR d,s		Bitwise Exclusive OR Immediate Bitwise Exclusive OR Register			
Destination d	Source s	Size 0 Byte 1 Word	Mode	Opcode 0Fh	Operand nn

This instruction replaces the destination register with the exclusive OR of the destination register and the immediate operand or the source register.

If the mode bit is set (immediate) then the immediate data is used. If the mode bit is clear (register) then the source register is used.

The operands may be either byte or word. The zero flag is set if the result is zero.

ADDM d,nn ADDM d,s		Add modulo 255 Immediate Add modulo 255 Register			
Destination d	Source s	Size 0 Byte	Mode 0 Register 1 Immediate	Opcode 10h	Operand nn

This instruction adds the immediate data or source register to the destination register modulo 255. This means that if the sum is greater than or equal to 255 then 255 is subtracted from the sum. This means that the result is always in the range 0-254.

If the mode bit is set (immediate) then the immediate data is used. If the mode bit is clear (register) then the source register is used. This instruction only works on byte operands.

The zero flag is set if the result is zero. The carry flag is unaffected.

SUBM d,nn SUBM d,s		Subtract modulo 255 Immediate Subtract modulo 255 Register			
Destination d	Source s	Size 0 Byte	Mode 0 Register 1 Immediate	Opcode 11h	Operand nn

This instruction subtracts the immediate data or source register from the destination register modulo 255. This means that if the result is less than zero then 255 is added to it. This means that the result is always in the range 0-254.

If the mode bit is set (immediate) then the immediate data is used. If the mode bit is clear (register) then the source register is used. This instruction only works on byte operands.

The zero flag is set if the result is zero. The carry flag is unaffected.

Shift d,nn Shift d,s		Shift Immediate Shift Register			
Destination d	Source s	Size 0 Byte 1 Word 2 Long	Mode 0 Register 1 Immediate	Opcode 12h	Operand nn

This group of instructions shifts or rotates the destination register by a number of bits specified by the immediate data or source register.

The shift mechanism has a fairly general one bit shift mechanism and a restricted multibit shift mechanism.

The one bit mechanism can shift bytes, words and longs, left or right. The bit which is shifted out is shifted into the carry flag and the bit which is shifted in may be the msb, the lsb, the carry flag or zero.

The mnemonic for the shift is made up of the letter "S" followed by the first letter of the direction of the shift "L" or "R" followed by the first letter of the bit which is shifted in "M","L","C" or "Z". The shift mode is encoded into bits 8-10 of the operand as follows. The table shows equivalent meanings of the instruction.

Operand 8-10	Shift	Equivalent
0	SLZ	Shift Left, Multiply by two
1	SLC	Rotate left through carry
2	SLM	Rotate left
3	SLL	
4	SRZ	Shift Right, Unsigned divide by two
5	SRC	Rotate right through carry
6	SRM	Arithmetic shift right, Signed divide by two
7	SRL	Rotate right

The shift count is encoded in bits 0-1 of the operand if immediate or bits 0-1 of the source register if register. The shift count is one more than the value as follows

Bit 0-1	Shift Count
0	1
1	2
2	3
3	4

The multi-bit shift is a simple extension of the single bit shift. The carry flag is still set to the first bit to be shifted out and all the new bits which are shifted in are the same as for the single bit shift. This simple mechanism does not allow multi-bit rotates but does support the arithmetic operations. Here are the sensible multibit shifts and their equivalent meaning. (The less meaningful multi-bit shifts are still available however).

Operand 8-10	Shift	Equivalent
0	SLZ d,nn	Multiply by 2^nn
4	SRZ d,nn	Unsigned divide by 2^nn
6	SRM d,nn	Signed divide by 2^nn

MOV d,nn MOV d,s		Move Immediate Move Register			
Destination d	Source s	Size 0 byte 1 word 2 long	Mode 0 Register 1 Immediate	Opcode 13h	Operand nn

This instruction loads the destination register with the immediate data or source register. If the mode bit is set (immediate) then the immediate data is used. If the mode bit is clear (register) then the source register is used.

The MOV instruction can load bytes, words or longs depending on the size field in the instruction but it can only move longwords in register mode as the immediate data is only 16 bits.

IMACC d,r0,r14		Initial Multiply Accumulate			
Destination d	Source X	Size 1 word 2 long	Mode X	Opcode 14h	Operand X

The IMACC and MACC instructions multiply R0 by the bottom two bits of R14 and add the product to the destination register. Register R14 is then shifted right by two bits and register R0 is shifted left by two bits.

By using a series of these instructions it is possible to multiply an N bit number in R14 by an M bit number in R0 and add the result to the destination register. The IMACC instruction should precede a series of MACC instructions.

R0 and the destination register may be word or long registers, as specified by the size field, R14 is always treated as a word.

The number of IMACC/MACC instructions needed is half the number of bits in R14. However the multiplication mechanism is intrinsically signed. This may mean that more bits are required to represent a value. For instance a value in the range 0-255 must be represented by a nine (or ten) bit twos complement number which requires five MACC instructions.

The multiplication mechanism uses the carry flag for its own purposes. The carry flag should be conserved from one IMACC/MACC to the next. The carry flag does not represent the carry out of the final accumulate.

The following examples show what the multiplier can do.

```
4 bit x 12 bit MACC
        IMACC.W   R2,R0,R14   ; +7 to -8 in R14
        MACC.W    R2,R0,R14   ; +2047 to -2048 in R0

8 bit x 24 bit MACC
        IMACC.L   R4,R0,R14   ; +127 to -128 in R14
        MACC.L    R4,R0,R14   ; +8M to -8M in R0
        MACC.L    R4,R0,R14
        MACC.L    R4,R0,R14

8 bit x 8 bit MACC
        IMACC.W   R2,R0,R14   ; +127 to -128 in R14
        MACC.W    R2,R0,R14   ; +127 to -128 in R0
        MACC.W    R2,R0,R14
        MACC.W    R2,R0,R14

16 bit x 16 bit MACC
        IMACC.L   R4,R0,R14   ; +32767 to -32768 in R14
        MACC.L    R4,R0,R14   ; +32767 to -32768 in R0
        MACC.L    R4,R0,R14
        MACC.L    R4,R0,R14
        MACC.L    R4,R0,R14
        MACC.L    R4,R0,R14
        MACC.L    R4,R0,R14
        MACC.L    R4,R0,R14
```

MACC d,r0,r14		Multiply Accumulate			
Destination d	Source X	Size 1 word 2 long	Mode X	Opcode 15h	Operand X

The MACC instruction is described with the IMACC instruction.

ELOAD d,(s)		Load Extended			
Destination d	Source s	Size 0 Byte 1 Word	Mode 0	Opcode 16h	Operand X

The ELOAD instruction loads the destination register from the memory location identified by the 32 bit source register.

ELOAD will load a byte or a word depending on the size field. The instruction timing is the same as for LOAD at the same address. ELOAD uses a 32 bit register instead of a 16 bit address in conjunction with the page register. This is useful if the processor needs access to data outside the 48k addressable within the current page.

There is no direct mode or modular addressing version of ELOAD

ESTORE (s),d		Store Extended			
Destination d	Source s	Size 0 Byte 1 Word	Mode 0	Opcode 17h	Operand X

The ESTORE instruction stores the destination register in the memory location identified by the 32 bit source register.

ELOAD will store a byte or a word depending on the size field. The instruction timing is the same as for STORE at the same address. ESTORE uses a 32 bit register instead of a 16 bit address in conjunction with the page register. This is useful if the processor needs access to data outside the 48k addressable within the current page.

There is no direct mode or modular addressing version of ESTORE

GMUL d,nn GMUL d,s		Galois multiply Immediate Galois multiply Register			
Destination d	Source s	Size 0 Byte	Mode 0 Register 1 Immediate	Opcode 18h	Operand nn

This instruction multiplies two numbers as if they were members of the Galois Field used by CD error control.

INTERRUPTS

There are eight interrupt sources Interrupt 0 to Interrupt 7. Each has an associated interrupt vector 10h, 20h 30h up to 80h. The interrupts are used as follows

Interrupt	Vector	Source
Interrupt 0	10h	CD sample rate or v32 input
Interrupt 1	20h	Host diagnostic register
Interrupt 2	30h	Byte channel end of transfer
Interrupt 3	40h	Change on pin v46
Interrupt 4	50h	Change on pin v47
Interrupt 5	60h	Timer overflow
Interrupt 6	70h	I2S Word select
Interrupt 7	80h	Word channel end of transfer
Interrupt 8	xxh	Resync

Each interrupt can be individually enabled or disabled by a bit in the interrupt mask register. Bit 0 corresponds to interrupt 0, bit 1 corresponds to interrupt 1 and so on. A one in the register enables the corresponding interrupt. The mask works at the input stage of the interrupt mechanism. If there is a pending interrupt from a given interrupt source then clearing the corresponding bit in the mask register will not clear the pending interrupt. The pending interrupt can only be cleared by vectoring to the corresponding interrupt service routine.

If a bit in the interrupt mask register is set then any interrupts which occurred while the bit was clear will be ignored.

All interrupts are enabled and disabled by the EI and DI instructions.

The interrupts are prioritised such that lower interrupts have higher priority.

Interrupts are automatically disabled on entry to the interrupt service routine and re-enabled by the return from interrupt instruction RETI. Interrupt latency is adversely affected by time spent with interrupts disabled. The programmer should re-enable interrupts in interrupt service routines as soon as possible. The RETI instruction should still be used to return from the interrupt even though interrupts have been enabled because RETI restores the flags whereas RET does not.

DMA CHANNEL

The DMA channel, transfers blocks of up to 64k bytes into memory byte at a time. This channel is intended for transferring CD data into memory.

The DMA channel has an address register, a count register and a page register. The address register is loaded with the bottom 16 bits of the start address of the data buffer. The page address is loaded with the next 16 bits of the address. The count register counts up so it should be loaded with 65536 - block size for the byte channel and 32768 - block size for the word channel.

The DMA mechanism uses "spare" memory slots. These are cycles which are not used by the processor for LOAD or STORE instructions etc. DMA latency and bandwidth will be adversely affected by long strings of such instructions. The DMA channel has a real time requirement that the latency be less than 3.9us for a single speed drive and correspondingly less for faster drives. Fortunately this translates to a fairly large number of instructions so allowing "spare" slots should not be onerous.

The DMA address and count registers are unbuffered so they must be reloaded in software. This is especially important for the byte channel where the DMA mechanism must be ready for the next frame. The end of transfer interrupt handler can be used to re-load these registers.

ERROR CORRECTION AND DETECTION

CD error detection and correction is performed in two stages, using separate parity words for the two processes. These processes are illustrated in Figs. 15 and 16.

The first process, C1, operates on a block 110 of 32 data bytes, which includes 4 parity bytes. This allows a maximum of 2 errors per block to be corrected.

The first stage is to calculate the syndromes for the block 110. The syndromes are cleared 112 using the NVP *SynClr* instruction. The block 110 is read 114 and each data byte in the block is added 116 to the syndrome using the *Check* instruction. Using one instruction per byte, dedicated hardware within the NVP processor calculates the syndrome according to the following formula:

$$\begin{aligned} syn0 &= byte0 * alpha\hat{}0 + byte1 * alpha\hat{}0 + byte2 * alpha\hat{}0 \ldots \\ syn1 &= byte0 * alpha\hat{}2 + byte1 * alpha\hat{}1 + byte2 * alpha\hat{}0 \ldots \\ syn2 &= byte0 * alpha\hat{}4 + byte1 * alpha\hat{}2 + byte2 * alpha\hat{}0 \ldots \\ syn3 &= byte0 * alpha\hat{}6 + byte1 * alpha\hat{}3 + byte2 * alpha\hat{}0 \ldots \end{aligned}$$

where alpha and powers of alpha are polynomials from the Galois field generated from the error correction polynomial by dedicated hardware within the NVP processor.

This process is repeated 118 and once the last data byte has been has been added to the syndrome, the NVP checks the syndromes 120,122 to see if they are zero, by reading the SynLow and SynHigh registers.

If the syndromes are zero, the data block does not have any errors and is marked as correct 124. Otherwise there is at least one error.

The number of errors is checked 126. If the data has only one error, there is a simple relationship between the syndromes which is easily detected. In this case, the error is corrected using the values of the syndromes and the data is marked as correct 128.

If the data has more than one error, the software may correct up to two errors providing that two error correction is enabled 130. If it is not enabled, then the software cannot correct more than one error and the data is uncorrectable. The data block is marked as bad 132.

If two error correction is enabled at C1 130, then the software will try to correct two errors. The first stage is to generate an error location polynomial from the syndromes. The order of the error location polynomial will indicate the number of errors in the data 134. If the order is greater than two, there are too many errors to be corrected and the data is marked as bad 132. If the order is two (one error will be dealt with at a previous stage), the error correction polynomial will be solved iteratively. The errors are then corrected 136. This process is described in detail by Lin and Costello, Error Control Coding, published by Prentice Hall. Once the errors are corrected, the data is marked as good. This correction process involves a large amount of arithmetic based on Galois field theory, which is supported by Galois arithmetic instructions in the NVP instruction set.

The second process, C2 error detection and correction, is shown in Fig. 16 and is similar to the C1 error detection and correction process, except that the position of the errors is already known from the C1 process and this allows up to four errors to be corrected. The first stage is to calculate and check the syndromes in the data block 140. This is the same process as that described above for the C1 process and will not be described in detail. Steps 142 to 150 of the process illustrated in Fig. 16 are therefore equivalent to steps 112 to 120 of Fig. 15. The data block is reduced to 28 bytes as four parity bytes have been removed as part of the C1 process. Of the 28 bytes, four are parity bytes, leaving 24 data bytes in the block.

If the syndromes are all zero 152, the data does not contain errors and is marked as good 154. Otherwise, the number of bytes of data that have been marked as bad are counted 156. As C1 and C2 corrections are orthogonal, only one byte from a C1 block is used for C2. Therefore C2 has one byte from each of 28 different C1 blocks. If there are more than four bad data markers, C2 error correction cannot be used. The error flags from C1 are propagated with the data 158. Otherwise, the data is corrected 160 using the position of the bad data flags to mark the error locations. This process is described by Lin and Costello. Once the data is corrected, it is marked as good 154. The correction process is supported by Galois instructions in the NVP instruction set.

CLAIMS

1. An integrated control and decoding circuit for a CD player, said circuit including digital input means for receiving a plurality of digital input signals including an input data signal containing input data read from a compact disk playing in said CD player and an input control signal containing instructions for controlling operation of the CD player; signal processor means for decoding and error correcting said input data; control means for controlling operation of said player in response to said input control signal; and digital output means for providing a plurality of digital output signals including an output data signal containing output data derived from said decoded and error corrected input data and an output control signal containing instructions for controlling operation of the CD player; wherein said digital input means, signal processor means, control means and digital output means comprise digital electronic components that are integrated onto a single semiconductor integrated circuit chip.

2. An integrated control and decoding circuit according to claim 1, including an analogue input interface having a plurality of analogue input ports for receiving analogue input signals and a plurality of analogue-to-digital converters (ADCs) connected to said analogue input ports, the digital output terminals of said ADCs being connected to said digital input means.

3. An integrated control and decoding circuit according to claim 2, wherein said ADCs are Sigma-delta ADCs.

4. An integrated control and decoding circuit according to claim 2 or claim 3, wherein said analogue input interface is integrated onto said single semiconductor integrated circuit chip.

5. An integrated control and decoding circuit according to any one of the preceding claims, wherein at least one digital output signal of said digital output means is modulated to provide an analogue drive signal.

6. An integrated control and decoding circuit according to claim 5, wherein at least one digital output data signal is modulated.

7. An integrated control and decoding circuit according to claim 5 or claim 6, wherein at least one digital output control signal is modulated.

8. An integrated control and decoding circuit according to any one of the preceding claims, wherein said signal processor means comprises a RISC processor that decodes and error corrects said input data under software control.

9. An integrated control and decoding circuit according to claim 8, including a hardware error checking and correction accelerator for calculating the syndrome of a block of data.

10. An integrated control and decoding circuit according to claim 9, in which the syndrome is calculated by Galois field arithmetic, the arithmetic process being conducted by a plurality of XOR gates in combination with look-up tables.

11. An integrated control and decoding circuit according to any one of the preceding claims, including a digital Phase Locked Loop (PLL) to lock to the CD data stream.

12. A CD player comprising a CD drive, a CD pickup assembly and an integrated control and decoding circuit, said control and decoding circuit including digital input means for receiving a plurality of digital input signals including an input data signal containing input data read from a compact disk playing in said CD player by said CD pickup assembly and an input control signal containing instructions for controlling operation of the CD player; signal processor means for decoding and error correcting said input data; control means for controlling operation of said player in response to said input control signal; and digital output means for providing a plurality of digital output signals including an output data signal containing output data derived from said decoded and error corrected input data and an output control signal containing instructions for controlling operation of the CD drive and pickup assembly; wherein said digital input means, signal processor means, control means and digital output means comprise digital electronic components that are integrated onto a single semiconductor integrated circuit chip.

13. A CD player according to claim 12, including an analogue input interface having a plurality of analogue input ports for receiving analogue input signals and a plurality of analogue-to-digital converters (ADCs) connected to said analogue input ports, the digital output terminals of said ADCs being connected to said digital input means.

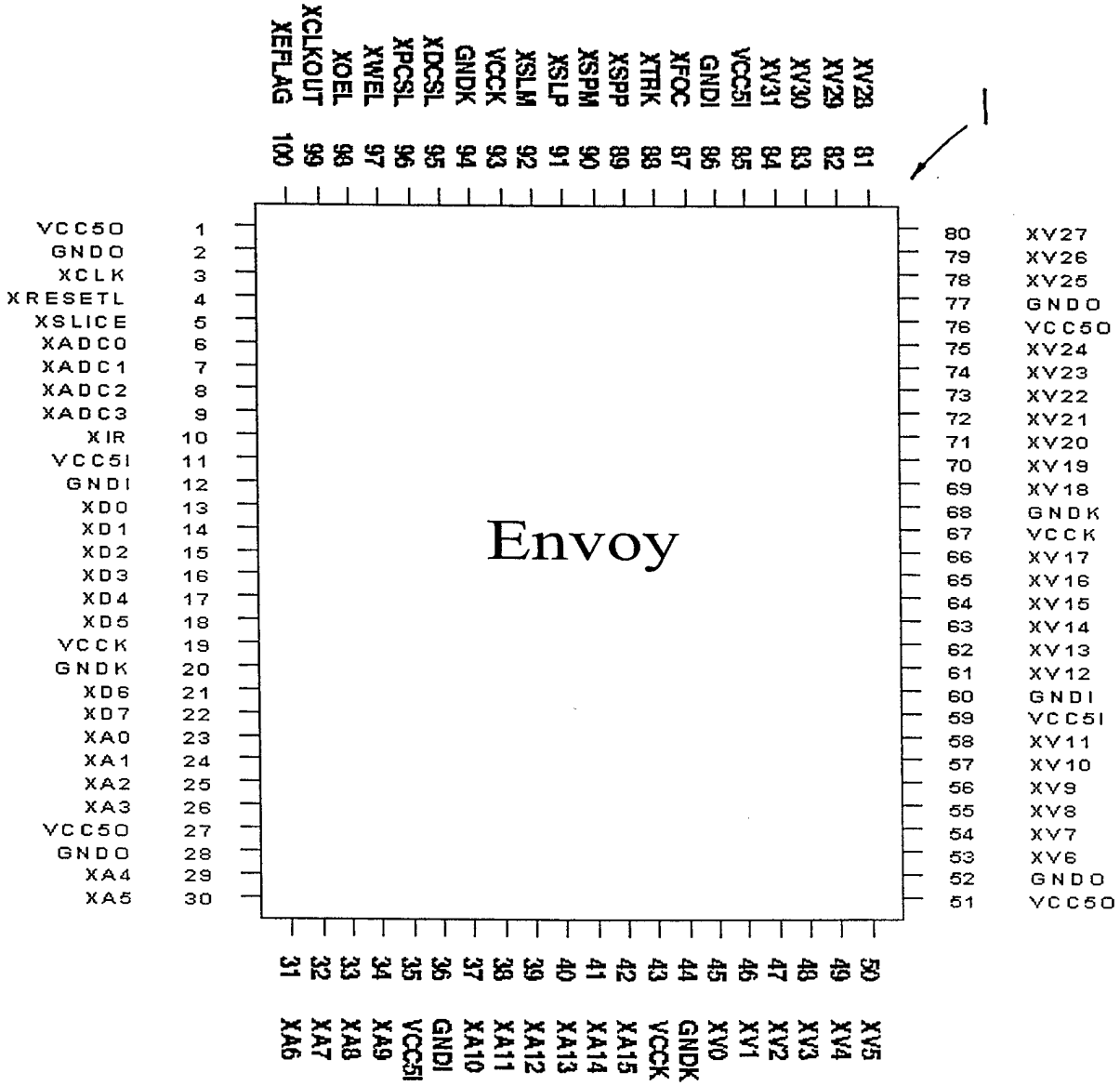

Envoy
VCC5O 1
GNDO 2
XCLK 3
XRESETL 4
XSLICE 5
XADC0 6
XADC1 7
XADC2 8
XADC3 9
XIR 10
VCC5I 11
GNDI 12
XD0 13
XD1 14
XD2 15
XD3 16
XD4 17
XD5 18
VCCK 19
GNDK 20
XD6 21
XD7 22
XA0 23
XA1 24
XA2 25
XA3 26
VCC5O 27
GNDO 28
XA4 29
XA5 30
31 XA6
32 XA7
33 XA8
34 XA9
35 VCC5I
36 GNDI
37 XA10
38 XA11
39 XA12
40 XA13
41 XA14
42 XA15
43 VCCK
44 GNDK
45 XV0
46 XV1
47 XV2
48 XV3
49 XV4
50 XV5
51 VCC5O
52 GNDO
53 XV6
54 XV7
55 XV8
56 XV9
57 XV10
58 XV11
59 VCC5I
60 GNDI
61 XV12
62 XV13
63 XV14
64 XV15
65 XV16
66 XV17
67 VCCK
68 GNDK
69 XV18
70 XV19
71 XV20
72 XV21
73 XV22
74 XV23
75 XV24
76 VCC5O
77 GNDO
78 XV25
79 XV26
80 XV27
81 XV28
82 XV29
83 XV30
84 XV31
85 VCC5I
86 GNDI
87 XFOC
88 XTRK
89 XSPP
90 XSPM
91 XSLP
92 XSLM
93 VCCK
94 GNDK
95 XDCSL
96 XPCSL
97 XWEL
98 XOEL
99 XCLKOUT
100 XEFLAG

Fig. 1

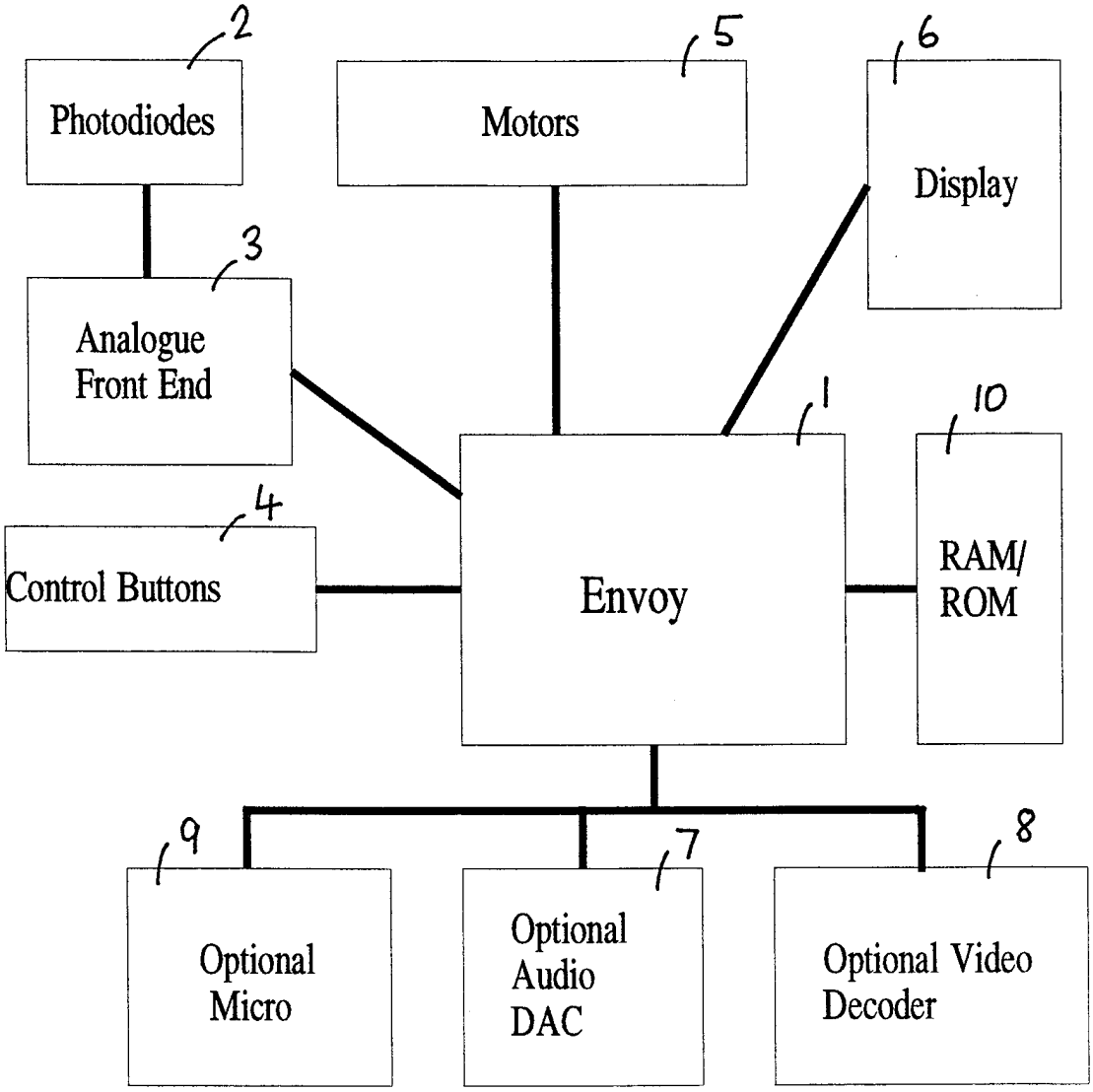

2
Photodiodes
5
Motors
6
Display
3
Analogue
Front End
1
10
4
Control Buttons
Envoy
RAM/
ROM
9
7
8
Optional
Micro
Optional
Audio
DAC
Optional Video
Decoder

Fig 2

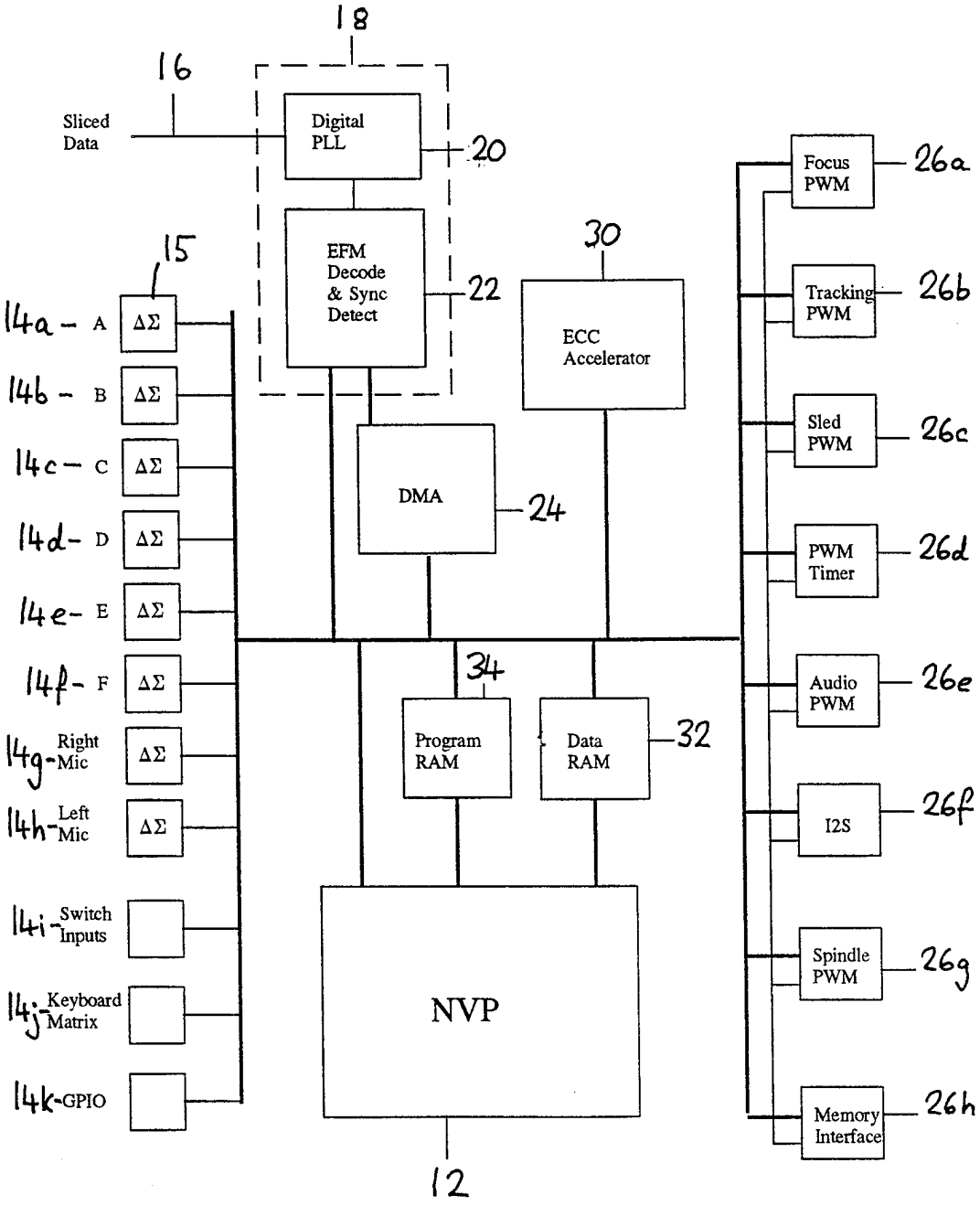
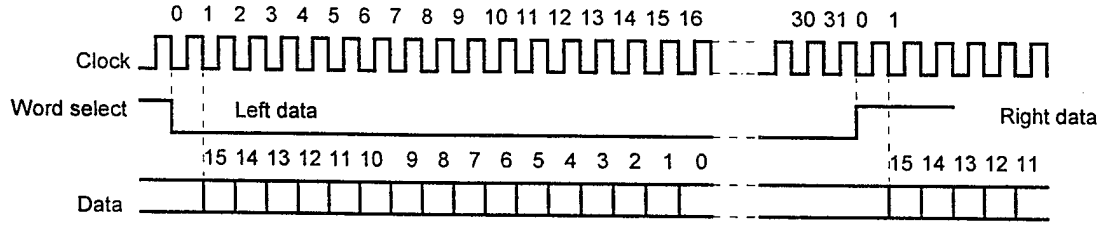
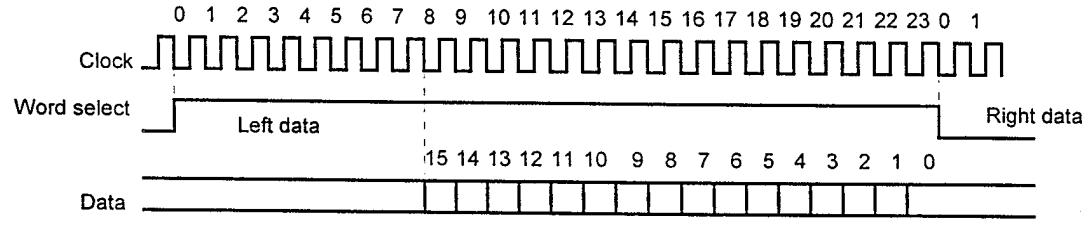

18
16
Sliced Data
Digital PLL
20
EFM Decode & Sync Detect
22
30
ECC Accelerator
15
14a – A
14b – B
14c – C
14d – D
14e – E
14f – F
14g – Right Mic
14h – Left Mic
14i – Switch Inputs
14j – Keyboard Matrix
14k – GPIO
ΔΣ
DMA
24
34
Program RAM
Data RAM
32
NVP
12
Focus PWM
26a
Tracking PWM
26b
Sled PWM
26c
PWM Timer
26d
Audio PWM
26e
I2S
26f
Spindle PWM
26g
Memory Interface
26h

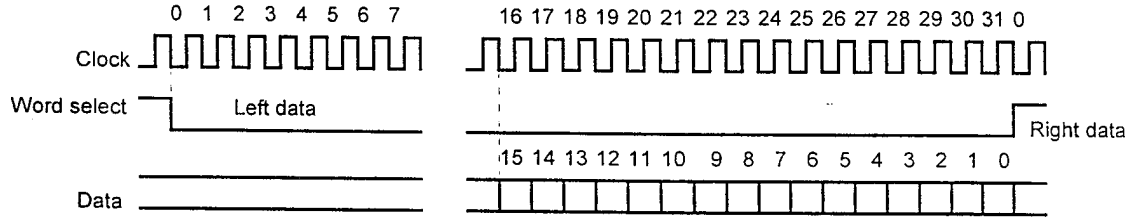
Fig 3.



Philips data format

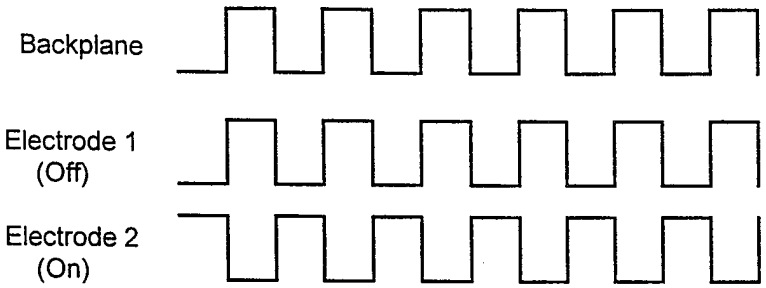


Sony data format



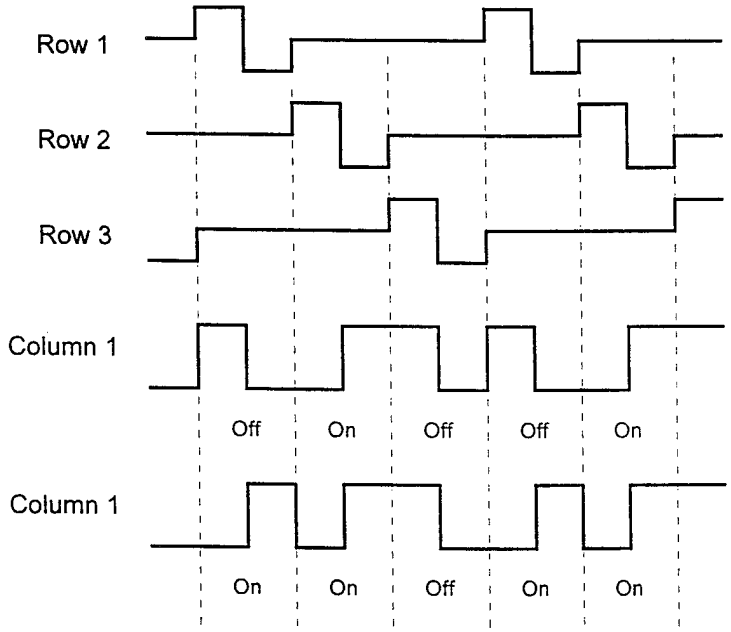
Matsushita data format

Fig. 4



Non Multiplexed LCD Drive

Fig. 5



3 Way Multiplexed LCD Drive

Fig. 6

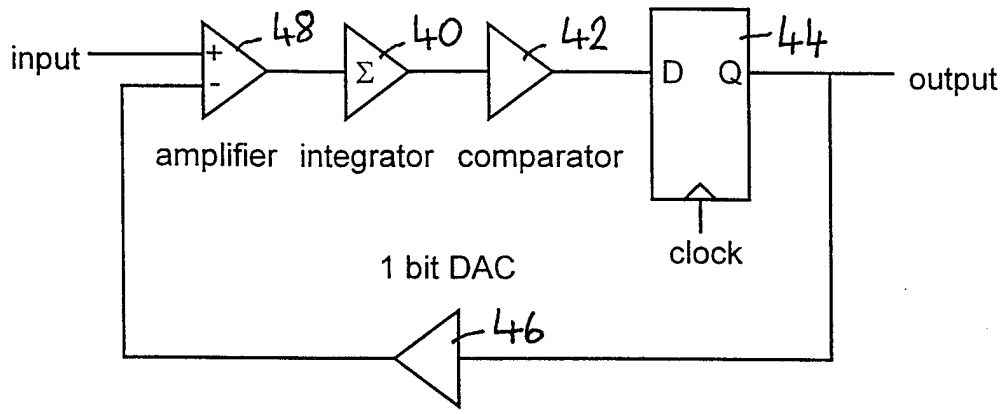

input
+
-
48
Σ
40
42
D
Q
44
output
amplifier
integrator
comparator
clock
1 bit DAC
46

Fig. 7

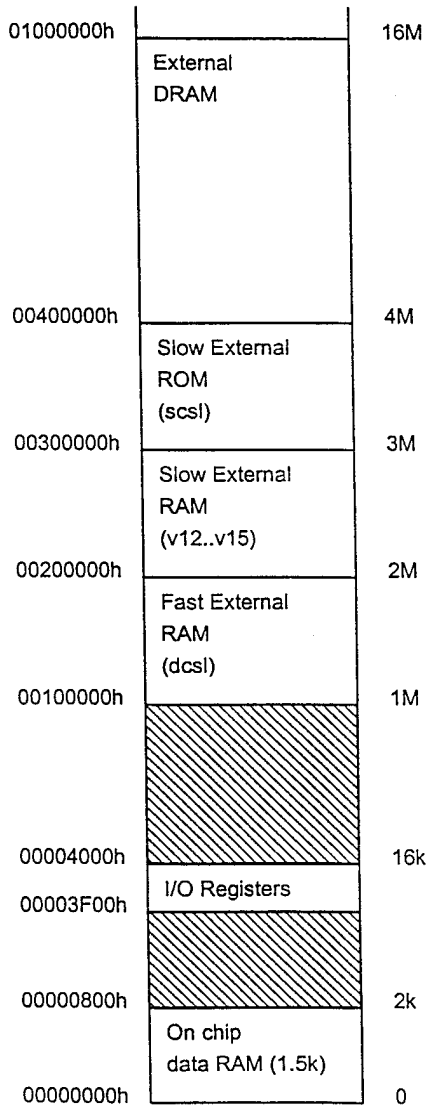
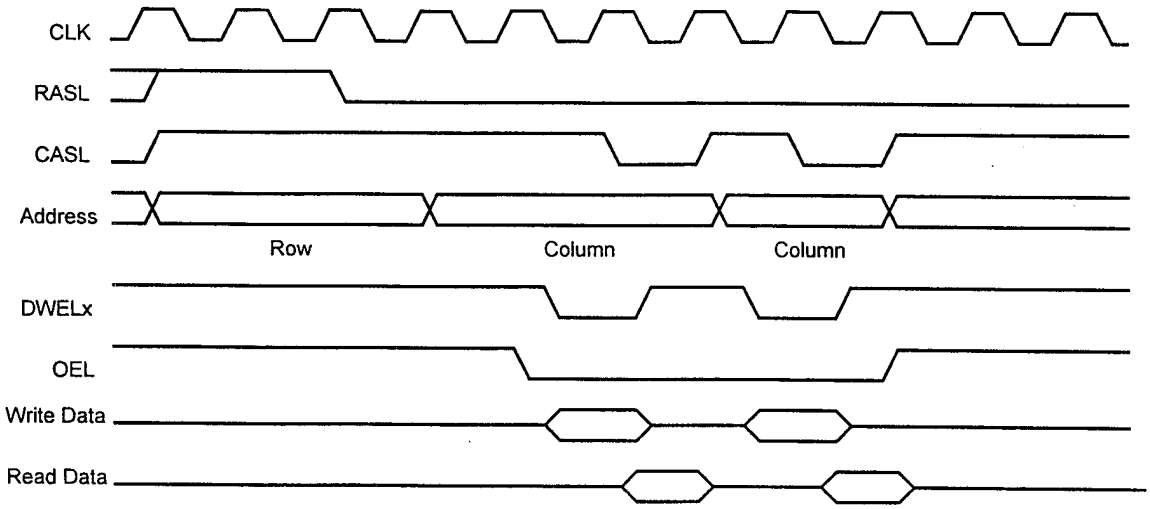
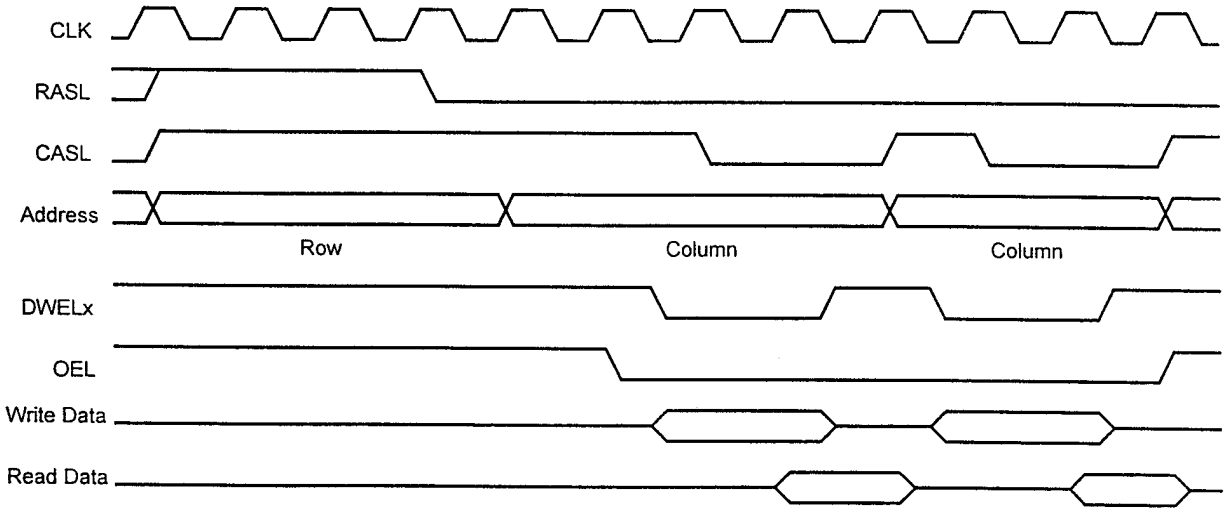

01000000h
16M
External DRAM
00400000h
4M
Slow External ROM (scsl)
00300000h
3M
Slow External RAM (v12..v15)
00200000h
2M
Fast External RAM (dcsl)
00100000h
1M
00004000h
16k
I/O Registers
00003F00h
00000800h
2k
On chip data RAM (1.5k)
00000000h
0

Fig. 8



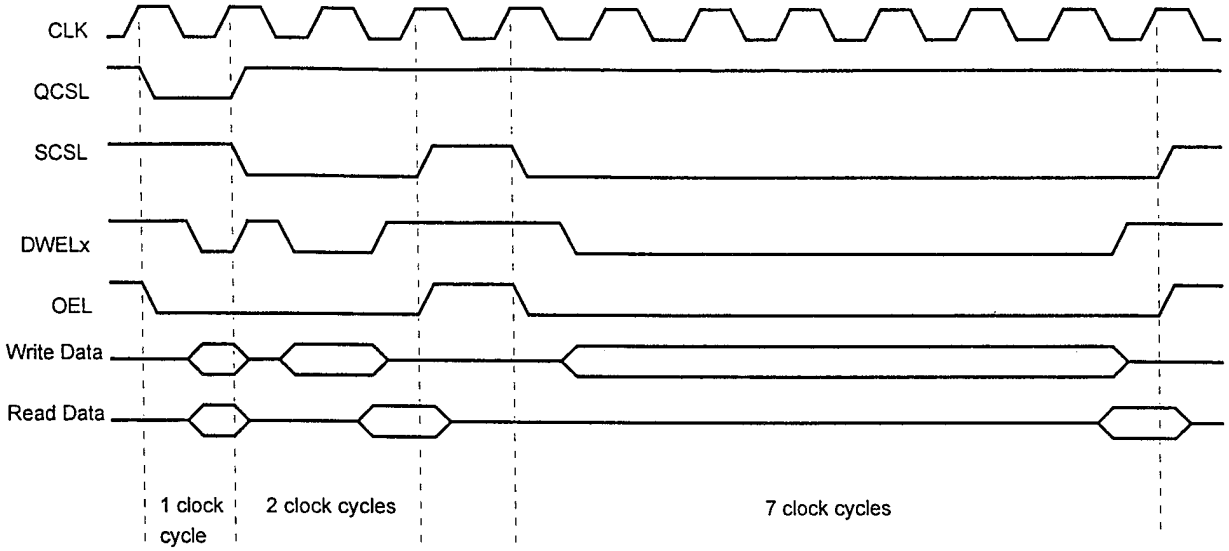
Fast DRAM RAS transfer followed by page mode transfer

Fig 9



Slow DRAM RAS transfer followed by page mode transfer

Fig 10



Fast and Slow Static RAM memory cycles

Fig 11

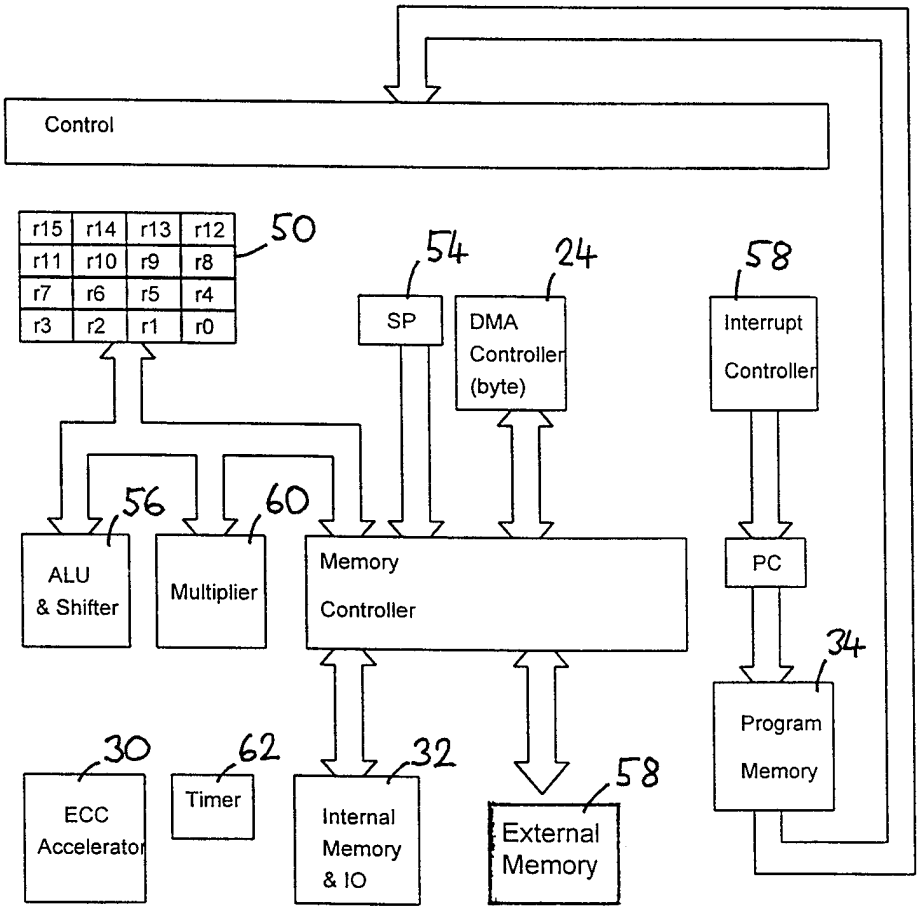


Fig 12 The NVP Architecture

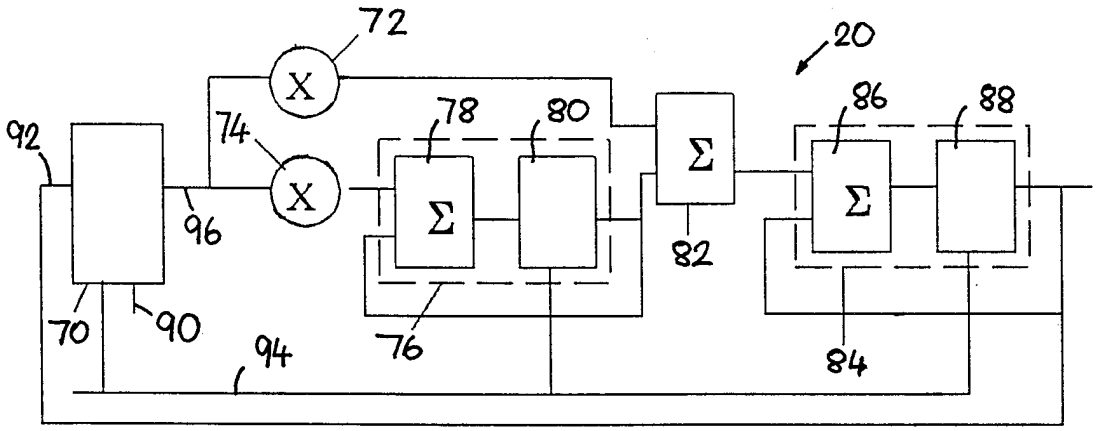

72
20
X
74
X
78
80
86
88
92
96
Σ
Σ
Σ
82
70
90
94
76
84

Fig. 13

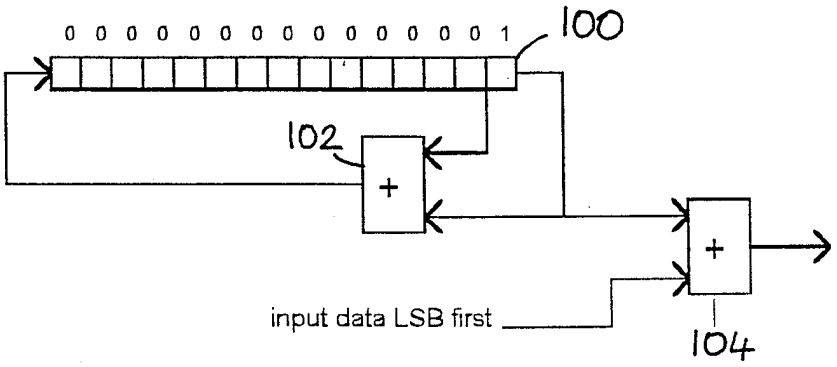

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
100
102
+
+
input data LSB first
104

Fig. 14

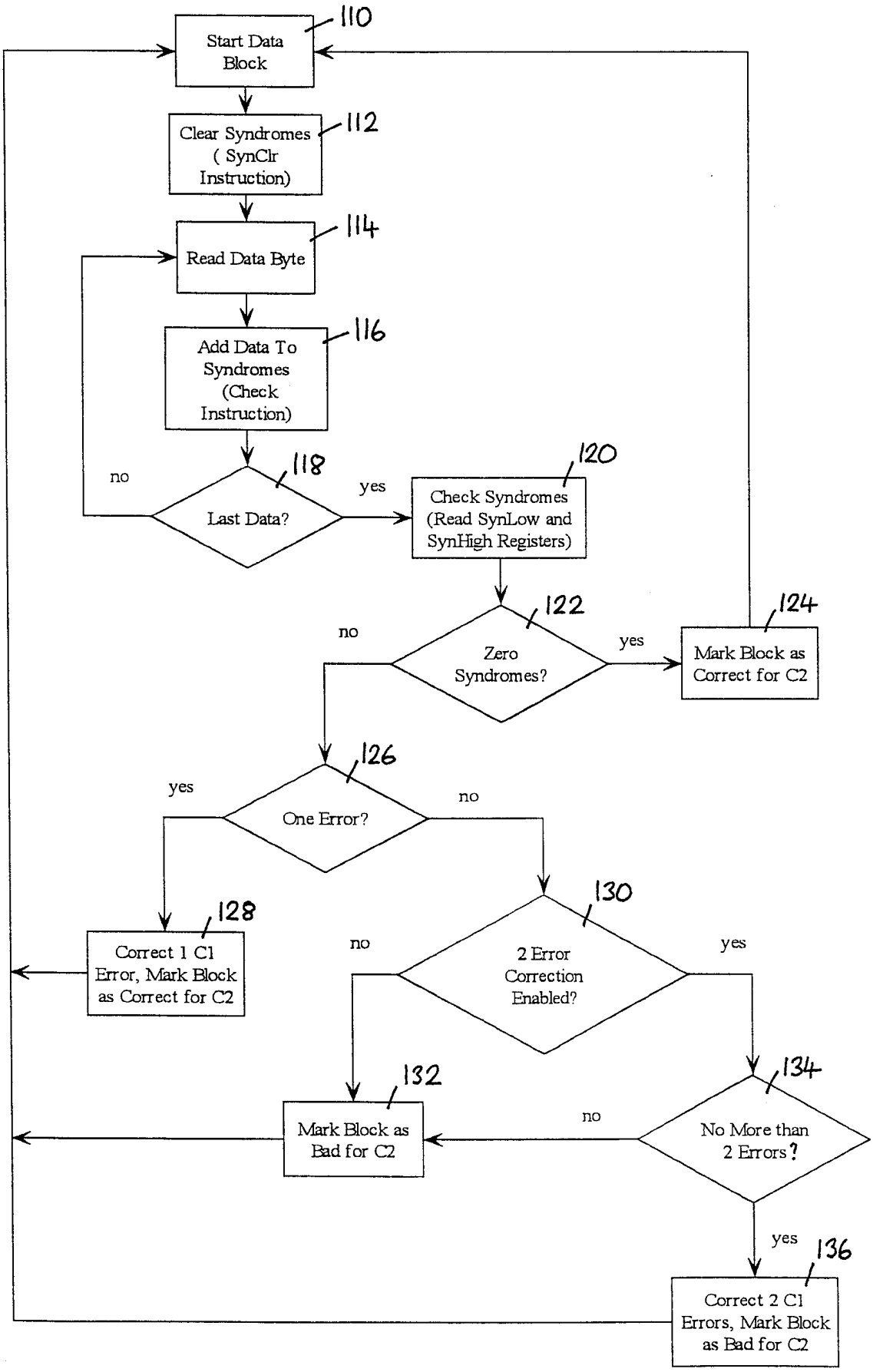

110
Start Data Block
112
Clear Syndromes (SynClr Instruction)
114
Read Data Byte
116
Add Data To Syndromes (Check Instruction)
118
Last Data?
no
yes
120
Check Syndromes (Read SynLow and SynHigh Registers)
122
Zero Syndromes?
no
yes
124
Mark Block as Correct for C2
126
One Error?
yes
no
128
Correct 1 C1 Error, Mark Block as Correct for C2
130
2 Error Correction Enabled?
no
yes
132
Mark Block as Bad for C2
134
No More than 2 Errors?
no
yes
136
Correct 2 C1 Errors, Mark Block as Bad for C2

Fig. 15

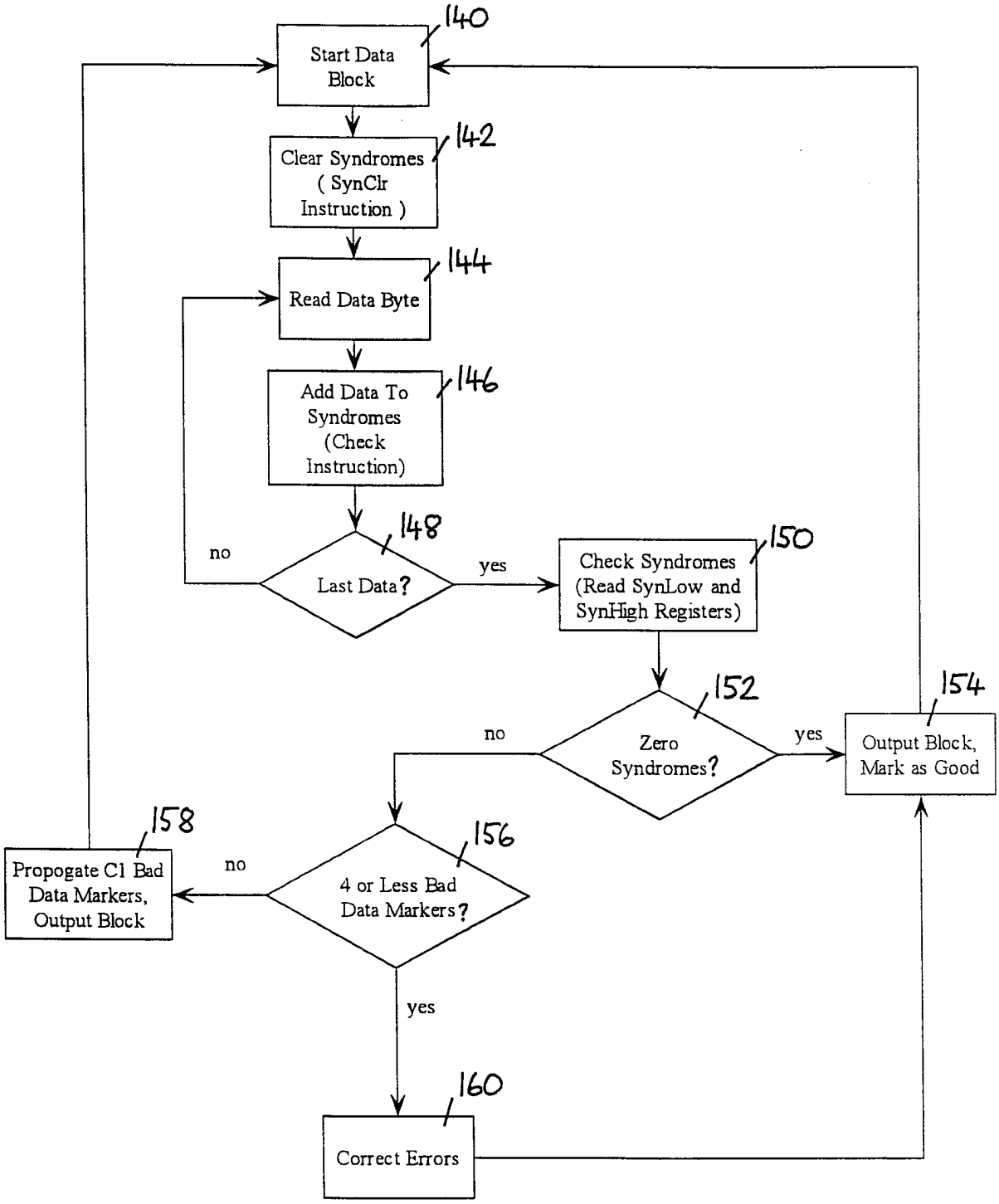

140
Start Data Block
142
Clear Syndromes (SynClr Instruction)
144
Read Data Byte
146
Add Data To Syndromes (Check Instruction)
148
Last Data?
no
yes
150
Check Syndromes (Read SynLow and SynHigh Registers)
152
Zero Syndromes?
no
yes
154
Output Block, Mark as Good
156
4 or Less Bad Data Markers?
no
yes
158
Propogate C1 Bad Data Markers, Output Block
160
Correct Errors

Fig. 16

INTERNATIONAL SEARCH REPORT

Inte ional Application No
PCT/GB 98/00747

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G11B20/00 G11B19/02 G11B20/18 G11B20/10

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G11B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 457 780 A (SHAW VENSON M ET AL) 10 October 1995 see the whole document	1,2,5-8, 12,13
X	EP 0 727 780 A (TOKYO SHIBAURA ELECTRIC CO) 21 August 1996 see the whole document	1,12
A	US 5 601 435 A (QUY ROGER J) 11 February 1997 see the whole document	1,2,8,12
A	US 5 600 844 A (SHAW VENSON M ET AL) 4 February 1997	
A	US 5 596 693 A (NEEDLE DAVID L ET AL) 21 January 1997	

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 August 1998	26/08/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Devergranne, C

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inte ional Application No
PCT/GB 98/00747

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication,where appropriate. of the relevant passages	Relevant to claim No.
A	US 5 398 139 A (BJEREDE BJOAN ET AL) 14 March 1995	
A	PATENT ABSTRACTS OF JAPAN vol. 096, no. 008, 30 August 1996 & JP 08 096512 A (TOSHIBA MICROELECTRON CORP;TOSHIBA CORP), 12 April 1996 see abstract	

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int ional Application No
PCT/GB 98/00747

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5457780 A	10-10-1995	AU 677791 B	08-05-1997
		AU 4768693 A	31-01-1994
		GB 2284525 A,B	07-06-1995
		WO 9401824 A	20-01-1994
		US 5611038 A	11-03-1997
		US 5754766 A	19-05-1998
EP 0727780 A	21-08-1996	DE 727780 T	13-02-1997
		JP 8287613 A	01-11-1996
US 5601435 A	11-02-1997	AU 693299 B	25-06-1998
		AU 4145696 A	31-05-1996
		EP 0789899 A	20-08-1997
		WO 9614627 A	17-05-1996
US 5600844 A	04-02-1997	CA 2147164 A	28-04-1994
		US 5745758 A	28-04-1998
		AU 684456 B	18-12-1997
		AU 3349993 A	09-05-1994
		GB 2286314 A,B	09-08-1995
		WO 9409595 A	28-04-1994
US 5596693 A	21-01-1997	NONE	
US 5398139 A	14-03-1995	NONE	