PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G05B 19/05 — A1

(11) International Publication Number: **WO 00/55698**

(43) International Publication Date: 21 September 2000 (21.09.00)

(21) International Application Number: PCT/US00/06796

(22) International Filing Date: 15 March 2000 (15.03.00)

(30) Priority Data:

60/124,500	15 March 1999 (15.03.99)	US
09/526,115	15 March 2000 (15.03.00)	US

(71) Applicant: SIEMENS ENERGY & AUTOMATION, INC. [US/US]; 3333 Old Milton Parkway, Alpharetta, GA 30005–4437 (US).

(72) Inventors: BOGGS, Mark, Steven; 901 Locklaine Drive, Johnson City, TN 37601 (US). FULTON, Temple, L.; 1508 Stateline Road, Elizabethton, TN 37643 (US). HAUSMAN, Steve; 2322 Camelot Circle, Johnson City, TN 37604 (US). MCNABB, Gary; Route 2, Box 155A, Unicoi, TN 37692 (US). MCNUTT, Alan; 2902 Newbern Drive, Johnson City, TN 37604 (US). STIMMEL, Steven, W.; 1223 Ridgeway Road, Johnson City, TN 37601 (US).

(74) Agents: ASPERAS, I., Marc et al.; Siemens Corporation, Intellectual Property Dept., 186 Wood Avenue South, Iselin, NJ 08830 (US).

(81) Designated States: CN, IN, JP, KR, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published

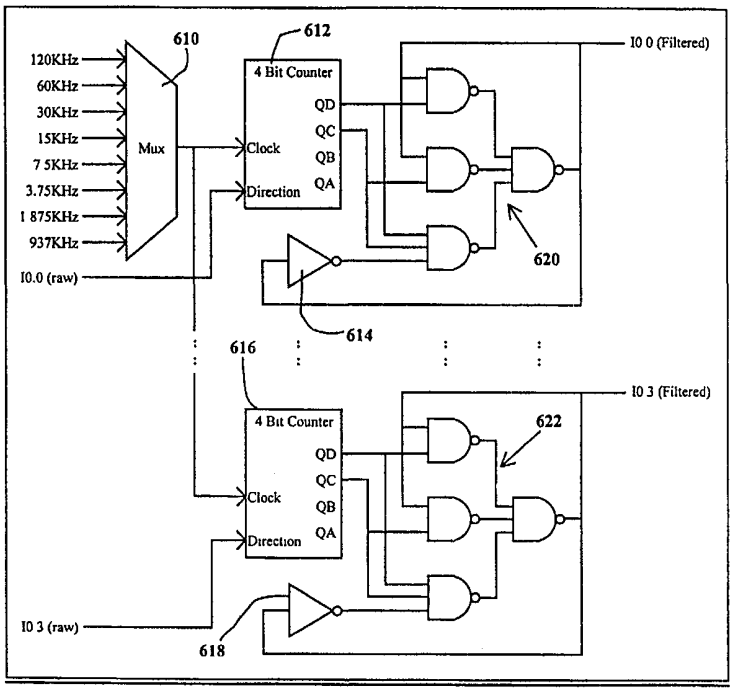
With international search report.

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: INPUT FILTER CIRCUIT FOR A PROGRAMMABLE LOGIC CONTROLLER AND ASSOCIATED METHOD

(57) Abstract

A programmable logic controller with enhanced and extended capabilities. A digital input filter implement filters with considerable less logic by simulating the action of a capacitor being driven by a constant current source whose output voltage is sensed by a comparator with a large amount of hysteresis. A pulse catch circuit captures the input pulse even though the update occurs between scan cycles. A pulse output controller includes a hardware pipeline mechanism to allow for smooth, hardware–controlled transitions from wave–form to wave–form. A free port link allows the user to control the port either manually or by operation of a user program. In order to provide higher performance for communication using PPI protocol, the PLC includes a built–in protocol. An n–bit modem protocol ensures data integrity without use of a parity type data integrity system. A hide instruction protects proprietary software by encrypting the sensitive code and decrypting the code during compilation and, thereafter, re–encrypting the code. A system function call allows the user to create and/or download new PLC functions and implement them as PLC operating system functions. An STL status function debugs programs during run–time and while the program is executing. A micro PLC arrangement provides compact size and efficiency.



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		

TITLE OF THE INVENTION

INPUT FILTER CIRCUIT FOR A PROGRAMMABLE LOGIC CONTROLLER AND ASSOCIATED METHOD

BACKGROUND

Field of the Invention.

The present invention relates to a programmable logic controller (PLC).

Related Information.

Programmable logic controllers (PLC's) are a relatively recent development in process control technology. As a part of process control, a PLC is used to monitor input signals from a variety of input points (input sensors) which report events and conditions occurring in a controlled process. For example, a PLC can monitor such input conditions as motor speed, temperature, pressure, volumetric flow and the like. A control program is stored in a memory within the PLC to instruct the PLC what actions to take upon encountering particular input signals or conditions. In response to these input signals provided by input sensors, the PLC derives and generates output signals which are transmitted via PLC output points to various output devices, such as actuators and relays, to control the process. For example, the PLC issues output signals to speed up or slow down a conveyer, rotate the arm of a robot, open or close a relay, raise or lower temperature as well as many other possible control functions too numerous to list.

The input and output points referred to above are typically associated with input modules and output modules, respectively. Input modules and output modules are collectively referred to as I/O modules herein. Those skilled in the art alternatively refer to such I/O modules as I/O cards or I/O boards. These I/O modules are typically pluggable into respective slots located on a backplane board in the PLC. The slots are coupled together by a main bus which couples

any I/O modules plugged into the slots to a central processing unit (CPU). The CPU itself can be located on a card which is pluggable into a dedicated slot on the backplane of the PLC.

Figure 36 shows one typical conventional programmable logic controller system as system 3610. System 10 includes a host programmable logic controller 3615 coupled by a field bus 3620 to a bus interface unit 3625. Bus interface unit 3625 couples and interfaces field bus 3620 to a local bus 3630 which includes a plurality of I/O terminal blocks 3635. I/O terminal blocks 3635 are coupled to respective I/O modules 3640 as shown in FIG. 1.

In system 3610, computational processing is performed by the host programmable logic controller 3615. In other words conditions are sensed at I/O modules 3640 and input data is derived therefrom. The input data is transferred through bus interface unit 3625 and field bus 3620 to host programmable logic controller 3615. Host programmable logic controller 3615 acts on the input data according to a control program stored in host PLC 3615. Host programmable logic controller 3615 processes the input data and produces output data in response thereto. The output data is transferred through field bus 3620, bus interface unit 3625, local bus 3630 to one or more I/O modules 3640. In response to the output data, the I/O module receiving the output data controls an output device coupled to the I/O module. I/O termination blocks are provided for coupling the I/O modules 3640 to the bus interface unit 3625.

The PLC may be arranged in a master/slave network as shown in FIG. 37a. In the figure, the master/slave control system includes a master PLC(M) and a plurality of remote slave units RSUs(R1-Rn). As shown therein, the master PLC(M) including a master PLC, a data link, and an I/O module, controls its own I/O connection points using a program and a communication parameter which are set by a user, and also controls the respective I/O connection points for the remote slave units R1-Rn. Each of the plurality of RSUs(R1-Rn) has at least one

I/O module, and carries out a data communication with the master PLC(M) through a communication cable, and accordingly controls its own I/O module. The RSUs may be PLCs acting as slaves.

With reference to FIG. 37b, the PLC and each of the RSUs include: a MODEM 3710 for carrying out a communication between the master PLC(M) and the RSUs via a communication cable; a receive/transmit module 3711 for exchanging data with the master PLC(M) according to a predetermined protocol; a receive/transmit buffer 3712 for temporarily storing therein the data for the exchange; an output data storage unit 3713 for storing therein the data which are to be transmitted from the master PLC(M) to an input/output module 3716; an input data storage unit 3714 for storing therein the data which are to be transmitted from the input/output module 3716 to the master PLC(M); and an input/output control module 3715 for controlling a data transmission between the data storage units 3713, 3714 and the input/output module 3716.

In operation, the data link in the master PLC(M) is a data linking device attached to the master PLC(M), which operates as a master unit in the related network, and which obtains an initiative of the data communication. The data link is able to set a maximum number N of RSUs. The RSU sets each number of its own and the master PLC(M), and receives a communication directly from the master PLC(M) for thereby carrying out a data transmission. The data link in the master PLC(M) sequentially selects the RSUs(R1-Rn) and carries out a data receiving/transmitting operation. For example, when the data outputted from the master PLC(M) is applied through the communication cable and the MODEM 3710 to the RSU(R1), the applied data passes through the receive/transmit buffer 3712 and the receive/transmit module 3711, and is stored in the output data storage unit 3713. The data stored in the output data storage unit 3713 is outputted to the input/output module 3716 in accordance with the control of the input/output control module 3715. The external control target data read from the input/output module 3716 of the remote slave unit R1 is stored in the input data

storage unit 3714 in accordance with the control of the input/output control module 3715. The data stored in the input data storage unit 3714 is transmitted through the receive/transmit buffer 3712 and the receive/transmit module 3711 to the master PLC(M).

The present invention provides new features that enhance and extend the capability of the conventional PLC.

OBJECTS AND SUMMARY OF THE INVENTION

It is an object of the present invention to enhance and extend the capability of the PLC.

It is another object of the invention to provide a digital input filter to enhance and extend the input capability of the PLC.

It is still another object of the invention to provide a pulse catch circuit to enhance and extend the pulse catching capability of the PLC.

It is yet another object of the invention to provide a pulse output controller to enhance and extend the output capability of the PLC.

It is a further an object of the invention to provide a free port link to enhance and extend the portability of the PLC.

It is still a further object of the invention to provide a protocol for modem communication to enhance and extend the connectivity of the PLC.

It is yet further an object of the invention to provide a hide instruction to enhance and extend the integration of the PLC with external programming applications.

It is still an additional object of the invention to provide a system function call to enhance and extend the function call capability of the PLC.

It is an additional object of the invention to provide an STL status to enhance and extend the status acquisition capability of the PLC.

It is yet and additional object of the invention to provide a micro PLC with an enhanced and extended capability.

In accordance with the foregoing objectives, the present invention provides a programmable logic controller with enhances and extended the capabilities.

In one aspect of the invention, a digital input filter is provided. The digital input filter simulates the action of a capacitor being driven by a constant current source whose output voltage is sensed by a comparator with a large amount of hysterisis. The digital filter implements input filters with considerably less logic.

In another aspect of the invention, a pulse catch circuit is provided. The pulse catch circuit captures the input pulse even though the update occurs between scan cycles.

In yet another aspect of the invention, a pulse output controller is provided. The pulse output controller smoothly transitions from one PTO or PWM wave-form to another. The pulse output controller includes a hardware pipeline mechanism to allow for smooth, hardware-controlled transitions from wave-form to wave-form.

In a still another aspect of the invention, a free port link is provided. The free port link allows the user to control the port either manually or by operation of a user program. In order to provide higher performance for communication using PPI protocol, a built-in protocol selection option is provided.

In still another aspect of the invention, a protocol for modem communication is provided. In a particular arrangement, the modem protocol supports communications over standard 10-bit, full duplex modems. The protocol uses a novel technique to ensure data integrity without use of a parity type data integrity system.

In a further aspect of the invention, a hide instruction is provided. The hide instruction provides for the protection of proprietary software by encrypting the sensitive code and decrypting the code during compilation and, thereafter, re-encrypting the code.

A still further aspect of the invention provides a system function call. The system function call allows the user to create and/or download new PLC functions and implement them as PLC operating system functions.

A yet further aspect of the invention is to provide an STL status function. The STL status function allows the user to debug programs during run-time and while the program is executing.

A quite further aspect of the invention is to provide the PLC in a micro PLC arrangement.

These and other objects of the invention will be readily understood from the following description of the figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a perspective view of the PLC of the present invention;
Figure 2 is a block diagram of the PLC;

Figure 3 is a schematic diagram of the PLC;
Figure 4 is a block diagram of the ASIC;
Figure 5 is a block diagram of the input block;
Figure 6a is a block diagram of the input filter circuit;
Figure 6b is a truth table of the input filter circuit;
Figures 6c and d are block diagrams of the input filter circuit;
Figure 7 is a table of delay times;
Figure 8 is a timing diagram of the scan cycle;
Figure 9 is a block diagram of the pulse catch circuit;
Figure 10 is a timing diagram of the scan cycle;
Figure 11 is a timing diagram of the scan cycle;
Figure 12a is a truth table for the pulse catch circuit;
Figure 12b is a table for the pulse catch circuit enable register;
Figure 12c is a table for the pulse catch circuit reserved registers;
Figure 12d is a table for the pulse catch circuit input point status register;
Figure 12e is a block diagram for the pulse catch circuit;
Figure 12f is a block diagram for the pulse catch circuit;
Figure 13 is a block diagram of the output block;
Figure 14a is a block diagram of the pulse output block;
Figure 14b is a table of registers of the pulse output block;
Figure 15 is a state diagram of the pulse output block;
Figure 16 is a software code listing;
Figure 17 is a state diagram of the pulse output block;
Figure 18 is a software code listing;
Figure 19 is a table for high speed operations;
Figure 20 is a block diagram of the I/O expansion slot.
Figure 21 is a table for the I/O expansion slot;
Figure 22 is a block diagram of the I/O expansion module;
Figure 23 is a table of components;
Figure 24 is a table of levels;
Figure 25 is a timing diagram of the read cycle;

Figure 26 is a timing diagram of the write cycle;

Figure 27 is a table of parity bits;

Figure 28 is a table of CPU types;

Figure 29 is a table of interrupts;

Figure 30a is a flow chart of the free port;

Figures 30b, c and d are tables of SM bit definitions

Figure 30e is a table of port definitions;

Figure 31 is a flow chart of the modem protocol;

Figure 32a is a table of control functions;

Figure 32b is a flow chart of the hide instruction;

Figure 33a is a table of control functions;

Figure 33b is a flow chart of the system function call;

Figure 34a is a table of STL instructions;

Figure 34b is a system diagram of the STL function;

Figure 34c is a flow chart of the STL function;

Figure 34d is a table of boolean expressions;

Figure 35a is a table of PLC parameters;

Figure 35b is a perspective view of the PLC and I/O expansion module;

Figure 36 is a block diagram of a PLC;

Figure 37a is a block diagram of a master/slave system; and

Figure 37b is a block diagram of communications in the master/slave.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

General Description Of The Programmable Logic Controller (PLC)

The exemplary Programmable Logic Controller (PLC) 100 of the present invention is shown in figure 1.

The figure illustrates the input/output (I/O) capabilities of the PLC 100. The user controls the PLC 100 by operation of the run/stop switch potentiometer and expansion I/O connector 104. Status light emitting diodes (LED's) 106 indicate the status of the PLC 100. A cartridge port 108 is provided for receiving a cartridge for expanding the function of the PLC 100 including, for example, increasing the memory. I/O LED's 110 are provided for indicating the status of the input/output pins of the PLC 100. A communications port 112 couples the PLC 100 to external components including, for example, other PLCs. The communications port may be used to connect the PLC 100 to other PLCs in a master slave relationship. The communications port 112 may also be used to connect to, for example, to a computer (such as a network computer or a personal computer). The PLC 100 may also connect to the internet through the communications port 112 via a modem or equivalent communications protocol device. User wiring connector 114 allows the user to connect external the PLC 100 to devices, such as motors and other peripheral devices.

In the figure, the PLC 100 is shown adjacent an expansion I/O that expands the input/output capabilities of the PLC 100. The expansion module, as well as the PLC 100, includes DIN rail mounting latch(es) 116 and a panel mounting location 118 for affixing the PLC 100 and the expansion I/O 102 to a suitable mounting fixture.

It will be appreciated that the PLC 100 of the present invention may include multiple arrangements and configurations of PLCs, several expansion I/O modules, and accessories that include memory cartridges, clock cartridges, battery cartridges cables or I/O simulator switch assemblies. This invention will address a fraction of the possible configurations relevant to the particulars of the various aspects described herein and the practitioner will instantly recognize these configurations with an understanding of the herein-described invention.

The PLC of the present invention is classified as a Micro PLC because of its small physical size. While the PLC is physically small in size, it includes many features that make it as powerful as (if not more powerful than) physically larger PLCs. Figure 1 shows a drawing of the PLC and an I/O module with possible dimensions. The design of the PLC 100 is described with reference to figures 39a and b.

Figure 2 shows a block diagram 200 of the PLC of the present invention. The block diagram shown is for a PLC with an integral AC power supply 202 for providing power. A central processing unit 210 is the core of the PLC 200 (100) and includes an ASIC, ROM and RAM. A digital input interface 212 is provided for inputing signals from the user wiring 114 (figure 1). A digital output interface 204 couples the PLC 200 to the user wiring 114 (figure 1). A communications interface 206 couples the PLC 100 to external devices via, for example, an RS-485 or token bus communication. An expansion I/O interface 208 connects the PLC 200 to expansion I/O interface(s) via, for example, a high speed, multiplexed bus.

In operation, the AC power supply 202 may provide an isolation boundary between the AC line and outputs, such as the 24VDC and 5VDC outputs. The PLC may be offered with a 24VDC power supply as well. The 24VDC power supply models would not provide isolation from 24VDC to 5VDC. The digital input interface 212 optically isolates the user wiring from the Central Processing Unit (CPU) 210 which includes all the logic level signals. The digital output interface provides a similar isolation boundary between the user wiring and the CPU. This isolation boundary is provided in the form of optical isolation or through relays where the relay coil is isolated from the relay contact. No isolation is shown in the figure between the CPU and the communication interface or the expansion I/O interface, but may be provided. All expansion I/O modules provide an optical isolation boundary between user wiring and the 5VDC logic signals. The CPU 210 is comprised of the ASIC which includes the

microprocessor, RAM and ROM. The ROM contains the operating system for the PLC 200 and may be either EPROM or FLASH EPROM depending upon the PLC model. The RAM is used for operating system data storage and scratch pad as well as for storing the user program that has been compiled into executable code.

A block diagram of the Central Processing Unit (CPU) 300 H/W is shown in figure 3. The connections of the CPU shown are not described in detail herein. The particular connections of interest here are the input circuits on I/O board 302, the I/O expansion bus 304 and the output circuit on I/O board 306 connections. In any event, figure 3 illustrates these details in sufficient schematic clarity that the practitioner will have no problem understanding the details from a review of the figure. It shall be appreciated that figure 3 illustrates only one arrangement and that those skilled in the art will readily understand how to implement other, equivalent, arrangements particularly in view of the instant specification.

The CPU 300 and connections are arranged as an ASIC shown in diagrammatic form in figure 4. As shown, the ASIC includes a microprocessor 406 (labeled "core"), address decode unit 402 and mapping logic. Two UARTs (Universal Asynchronous Receiver Transmitter) 424, 426 are provided that effect the electronic circuit that makes up the serial port. UARTs convert parallel bytes from the CPU into serial bits for transmission, and vice versa. However, other communication arrangements are possible. There are digital input conditioning circuits 418, high speed counters 410, pulse train output circuitry 420, potentiometer circuitry 414, watchdog circuitry 430 and reset circuitry 428. A bus interface 404 is provided. The expansion I/O unit is labeled 422. A test interface is provided for testing the PLC 100. The power distribution circuit is illustrated as circuit 432. The memory is shown as internal RAM 408. An interrupt control unit 412 handles interrupts for the CPU 406.

The PLC 100 provides a means for users to create specific application control programs that, when executed by the PLC, direct the operation of machines and/or processes used in the manufacturing of a wide variety of products. In this way the PLCs are similar to all other PLCs and the practitioner will immediately understand the more mundane aspects of the invention with such basic understanding of PLCs. In spite of the similarity in use and function of the PLCs there are many unique features and functions that are blended into the PLC 100 of the present invention which expand and enhance its utility to the user as will herein be described.

Digital Input Point Unit

The digital input point unit 418 will be described. After each of the digital input signals (I0.0 to I1.5) has been converted to a compatible signal, e.g., +5V signal, and isolated from the user wiring, it is fed into the Digital Input Point Unit 418 of the ASIC. A block diagram for input points IB0[2:0] which illustrates the functionality provided by the Digital Input Point Unit 500 (418, figure 4)is shown in figure 5.

This unit 500 performs a number of functions on the digital input points connected from the user's application to the CPU. This unit filters the input points and provides access to the filtered state of these points. A pulse catch functionality is provided on each input point to optionally allow the capture of short duration pulses. An edge interrupt function is provided to generate interrupts on the occurrence of rising and falling edge transitions on certain input points. High-speed counters are used to count events that occur too fast for the CPU scan rate.

As shown in figure 5, a digital input filter(s) 506_{1-n} provides a software configurable filter. That is, the PLC sets the filter parameters that control the filtering function in accordance with software executed by the PLC. In at least

one arrangement, up to 8 filter values are configurable by software, from 0.2 msec to 12.8 msec, for example. Filter implementation is possible with, for example, a 4-bit up/down counter on each input point. A pulse catch circuit(s) 508_{1-n} allows for the capture of a rising or falling edge transition of the filtered input until software has a chance to read the value. Independent enable/disable of the pulse catch function is provided for each input point. An edge interrupt circuit(s) 510_{1-n}, when this function is enabled, generates an interrupt upon the occurrence of a rising edge and/or falling edge transition on, for example, up to four input points. High-speed counter(s) 514 (516), in this case six high-speed counters (HSC) are supported, each contain, for example, a 32-bit up/down counter and a 32-bit compare register. The 32-bit register captures the counter value so that it can be read by software. Each counter and compare register are loadable by software. Four of the HSC's, for example, can handle two-phase clocking. The other two HSC's may support single-phase clocking only. Each HSC can generate interrupts upon the occurrence on equality of the counter vs compare register, upon a direction change condition, and upon the assertion of the HSC's external reset input.

Digital Input Filter Circuit

The implementation of the input filters in previous ASIC circuitry consumes too many gates. In the ASIC of the present invention, there is a need to provide compatible functionality without consuming so many gates. The following is a description with reference to figures 6a to 6d and 7 that implements input filters with considerably less logic in the instant ASIC by employing the truth table of figure 6b.

The ASIC provides digital input filtering for, for example, fourteen digital input points as shown in figure 6a and 6b. In figure 6a, the digital input filter circuit 600 (506, figure 5) includes input 602, input filter circuit 604, selected delay time

606 and output 608. The filter delay times set by the selected delay time 606 are software configurable to, for example, one of seven different values. The filter delay times are selected through settable registers internal to the ASIC. The following table in figure 7 describes the possible relationship between the value written to the register and the corresponding delay time selected. Of course, other relationships are possible.

Each digital input filter simulates the action of a capacitor being driven by a constant current source whose output voltage is sensed by a comparator with a large amount of hysteresis. The digital equivalence of this analog circuit is implemented as a four bit, up/down counter whose counting direction is controlled by the state of the input point. The frequency at which the counter is clocked is determined from the selection values written into registers internal to the ASIC by software. The 1 MHz Master Clock is used as the time base for each filter so that its operation is independent of the system clock frequency. The delay time is calculated from the following formula:

digital filter delay time = 12 * (period of the up/down counter's selected input clock)

The truth table in figure 6b defines the operation of the digital equivalent circuit where the result produced by the filter is the output value 608 (FIB*y.x*). One skilled in the art will immediately appreciate that the details of the circuitry may be implemented from the table in figure 6b using well-known boolean logic. The exemplary circuit is shown in figure 6c.

For each group of, for example, four inputs a multiplexer 610 outputs one of eight clocks which is used to set the input filter delay time. The clock selected by the multiplexer 610 drives a four bit up/down counter 612 (one for each of the inputs in each group of four inputs). The physical input controls the direction (up/down) of the four bit counter. When the input is ON (logical '1') the counter

counts up. When the input is OFF (logical '0') the counter counts down. The count up sequence of the counter is:

0, 1 , 2, ... 13, 14, 15, 15, 15, ... 15

The count down sequence of the counter is:

15, 14, 13, ... 2, 1, 0, 0, 0, ... 0

As indicated by the counting sequence, the counter will not roll over when counting up or when counting down.

The filter delay 614 is designed to mimic the operation of a constant current source driving a capacitor whose voltage is sensed by a comparator with a large amount of hysteresis. In this case, in order to detect an input transition from off to on, the counter must reach a count of 12. Once this count is reached, the output will be turned on and it will remain on until the threshold for an input transition from on to off is reached at a count of 3. The nand gate logic 620 shown in the figure implements these thresholds.

The thresholds are semetric in that 12 up counts or 12 down counts are required from a steady state low or high input condition. The table in figure 6d lists the delay times for 12 clocks at each of the frequencies supplied to the multiplexer. Of course, other delay times are possible. Additional layers, each layer including a counter 616, nand gate logic 622 and a delay 618 may be provided for each input that operate in a corresponding manner as the afore-described layer.

Pulse Catch Circuit

Following the digital input filter circuit is a pulse catch circuit (508, figure 5). The purpose of this circuit is to capture either a change in the input state from high to low or from low to high and hold it until the PLC operating system software has recognized the change in state. The PLC reads the state of the inputs in a cyclic fashion in both STOP and RUN modes. (The main difference between STOP and RUN modes in the PLC is that in STOP mode the user's program is not executed whereas in RUN mode the user's program is executed.) Figure 8 shows the PLC scan cycle (the cycle is normally called the PLC scan cycle or simple scan.) As shown in figure 8, the PLC scan cycle in RUN mode reads the input once per cycle. The same scan cycle is followed in STOP mode, but the user program is not executed in STOP mode.

Since inputs are only read once per cycle, it is possible for inputs to change state without the PLC ever recognizing the change. Such state changes that occur too fast for the PLC to recognize are referred to as pulses. In order to prevent the PLC from missing a short pulse on an input, the user can enable an input interrupt which will suspend normal program execution while the PLC services the interrupt. This method is very effective, but it requires additional support from the CPU and uses up considerable execution time to process the interrupt routine. For this reason only a small number of input interrupts are allowed.

The other method is to provide pulse catch circuits on each of the integral inputs to the PLC. This method allows infrequent but short duration pulses (either high going or low going) to be captured and held until the PLC recognizes the change at the appropriate point in the scan cycle.

Pulse Catch Circuit Top-Level Block Diagram

Now in more detail, figure 9 illustrates the pulse catch circuit 900 (508, figure 5) wherein input 902 are coupled to pulse catch circuit 904, are captured according to the enable signal 906, and output 908. In this manner, the pulse capture

circuit is able to capture and hold digital input point pulses whose duration is greater than the selected filter time but is less than the scan time. Figure 10 illustrates the problem where the pulse is missed because the pulse occurred between the input updates.

The CPU software is designed to read the state of all input points once per CPU scan and this operation is called the input update. As mentioned above, a pulse can be missed if it occurs between the input updates of consecutive CPU scans (see the example in figure 10). The pulse catch circuit 904 captures and holds this type of event. In one possible arrangement, there are fourteen pulse catch circuits, one for each input point.

When a pulse catch circuit is enabled, a change in state (low -> high or high -> low) of an input is captured. Further state changes of the input are ignored until the captured value has been read by the software during the next CPU scan. Once the captured value has been read, the circuit is then able to detect new input state changes.

The operation of the pulse catch circuit can be described more precisely with the truth table for a synchronous state machine shown in figure 12a. The symbols used in this table are defined below:

PCE - Pulse catch circuit enable bit

I - Input signal synchronized to the system clock by the input filter circuit (FIB*y.x*)

CV- The input value captured after state change of the input and after an input update

(synchronized to the system clock)

F - State change flag (synchronized to the system clock)

RP- Read pulse (one for each input status byte) generated by a read of the input status byte

(synchronized to the system clock in such a way that it is active for one clock cycle and goes

inactive on the clock edge that transferred the status of the input)

PS- Present State

NS- Next State

In operation, and with reference to figure 12a, when the circuit detects a change of state on the input, the new value of the input is captured and a state change flag is set. While the state change flag is set, the stored input value is held and cannot be updated. Therefore, any input changes occurring while the state change flag is set are ignored. After the stored value is read by software, the state change flag is reset, the current state of the input is captured, and the detection of input state changes is re-enabled. This sequence of actions is called re-triggering the pulse catch circuit. The result is that the input pulse is captured even though the update occurs between scan cycles as shown in figure 11. See, for example, CIB1.6 that shows that the pulse catch circuit output 908 captures the input pulse 902 even though the pulse occurs between scans, i.e., scan n and scan n+1.

Two 8-bit registers, for example, are provided in this block for individually enabling and disabling the pulse catch function on each of the sixteen input points. These enable registers are defined in the table shown in figure 12b. Input status registers are provided to give the CPU software access to the conditioned input point states, **CIB0[7:0]** and **CIB1[5:0],** which are output from the pulse catch circuitry. These are shown in figure 12c. The format of the input status registers is shown in figure 12d.

The circuit elements of the pulse catch circuit, as will be appreciated by those skilled in the art, will be readily recognizable from the truth table shown in figure 12a. It will further be appreciated that the pulse catch circuit, therefore, may have a plurality of configurations that fulfill the truth table.

The exemplary pulse catch circuit that operates in accordance with the foregoing description is shown in figures 12e and 12f. A system clock pulse drives flip-flops 1202, 1208. The input signal is coupled to the first flip flop 1202, the Q output of which is coupled to the nand gate logic 1206. Signals PCE, RP and F are delayed and coupled to corresponding nand gates in the nand gate logic 1206, as shown. In addition, the output of the second flip flop 1208 is fed back to the nand gate logic 1206. The output of the nand gate logic is coupled to the second flip flop 1208 and, upon being latched thereby, is clocked to the output. The F signal is generated in figure 12f, wherein the system clock drives flip flops 1210 and 1216. The first flip flop latches the input and the second flip flop latches the output of the nand gate logic 1214. The nand gate logic 1214 is arranged as shown to implement the afore-described logic on the signals output from the first flip flop (Q), the inverted version of Q, the captured version of the input signal and its delayed version (delayed by the delays 1212), the RP signal delayed by delays 1212 and the PCE signal.

The pulse catch circuits can be individually enabled and disabled. If a pulse catch circuit is disabled, then the circuit input is passed through to the circuit output. If enabled, then the pulse catch functions described above are active.

Digital Output Point Unit

This digital output unit (420, figure 4) allows direct software control of the state of each output point. In addition, a pulse output capability allows pulse train output and pulse width modulated output on up to two of the output points.

Now with reference to figure 13, the ASIC provides two output point state registers 1302, 1304 for software control of the state of ten digital output points. In addition, two Pulse Output Blocks 1306, 1308 (PLSs) may be implemented to

provide the capability of generating pulsed wave-forms at a frequency faster than software can accomplish.

Within the pulse output blocks 1306, 1308, the ASIC provides a choice of two output functions, e.g., PWM (pulse width modulated) or PTO (pulse train output). The PWM function provides for continuous pulse output with a programmable cycle time and duty cycle. Intended mainly for stepper motor applications, the PTO function provides for the output of a specified number of approximately 50% duty cycle pulses. Of course, other applications will be appreciated by those skilled in the art. The cycle time in the PTO function is also programmable and can either be fixed or can be automatically adjusted to increase or decrease at a user-specified rate.

When PLS0 is operating (enabled), it has control of output point **XQB0.0** via MUX 1310. Otherwise, **XQB0.0** is controlled by the value which software writes to the output point state register of the ASIC. Likewise, PLS1 controls the state of **XQB0.1** if enabled, via MUX 1312, and the output point state register of the ASIC controls the state of **XQB0.1** if PLS1 is not enabled.

Pulse Output Blocks

The ASIC shown supports two pulse output (PLS) blocks. Of course, any number of PLSs are configurable. These blocks allow the CPU to produce waveforms at frequencies faster than the CPU software is capable of manually generating. The output from Pulse Output Block 0 (PLS0) is one of the sources of output point **XQB0.0** and the output from Pulse Output Block 1 (PLS1) is one of the sources of output point **XQB0.1**.

The ASIC for the PLCs provided both the Pulse Train Output (PTO) and the Pulse Width Modulated (PWM) functions. PTO is an operational mode in which

the period (cycle time) of the wave-form and the number of cycles (pulse count) is specified by software. The duty cycle of the wave-form is approximately 50%. PWM is an operational mode in which the period (cycle time) and duty cycle (pulse width) are specified by software and the wave-form is output continuously.

It was discovered that there was a fundamental problem with the implementation of these functions in that there was no way to smoothly transition from one PTO or PWM wave-form to another. Each transition required that the pulse output block be stopped by the software and then restarted which introduced a discontinuity in the resultant output wave-form.

Because of the problem with transitions from one wave-form to another as described above, the ASIC of the present invention provides the pulse output block with a hardware pipeline mechanism to allow for smooth, hardware-controlled transitions from wave-form to wave-form. Also, to allow for better operation with stepper motors, the ability to automatically ramp the value of the cycle time at a specified rate was added to the ASIC.

PLS Block Diagram and Operational Overview

The block diagram in figure 14a shows the organization and components of each pulse output block 1400.

PLS Time Base Generation - The time base (e.g., 1ms or 1µsec) is provided by a multiplexer 1402 that selects either, for example, the 1MHz Master Time Base clock or a 1 KHz clock derived from the Master Time Base clock.

Wave-form Period Control - The output of the time base multiplexer drives a 16-bit up counter, called the cycle time counter 1420. This counter is incremented on the rising edge of the time base clock. The output of this counter is compared by comparator 1430 with the value stored in a 16-bit cycle

time preset register 1422 to control the period of the output wave-form. When the value of the cycle time counter reaches the value of the cycle time preset register, then the current cycle is complete and the *CycleDone* event is generated.

The values stored in the cycle time counter 1420 and cycle time preset register 1422 are unsigned values. The valid range of the cycle time counter 1420 is, for example, 0 to 65535. The valid range of the cycle time preset register is 2 to 65535. The comparison to determine *CycleDone* is an unsigned comparison.

A delta cycle time register 1406 exists for use in PTO mode to hold a signed, twos-complement 16-bit value that is added to the value stored in the cycle time preset register 1422 at each *CycleDone* event. If the sum exceeds 65535 or is less than 2, then the ***AdderError*** event is generated. For example,

preset = 65530 (FFFAH unsigned) delta = -20 (FFECH signed)	preset = 65530 (FFFAH unsigned) delta = 20 (0014H signed)
sum = 65530+(-20) = 65510 **OK**	sum = 65530+(20) = 65550 **Adder Error !!!**
preset = 100 (0064H unsigned) delta = -20 (FFECH signed)	preset = 100 (0064H unsigned) delta = -120 (FF88H signed)
sum = 100+(-20) = 80 **OK**	sum = 100+(-120) = -20 **Adder Error !!!**

In PWM mode, the delta cycle time register is held at a value of 0.

Software may load the cycle time preset register 1422 and the delta cycle time register 1406 only when the PLS block is disabled. The valid range of this register in PTO mode is, for example, -32768 to +32767.

Wave-form Duty Cycle Control - The output of the cycle time counter 1420 is also used in the control of the wave-form's duty cycle. There is an unsigned

comparison performed between this counter value and a pulse width preset value whose source is determined by the PLS operating mode. The state of the PLS block output signal **PLSxOUT** is determined by this comparison. While the counter value is less than the pulse width preset value, the **PLSxOUT** signal is high. When the counter value reaches or exceeds the preset, then the **PLSxOUT** signal is driven low. The pulse width preset value is obtained from the lower 16-bits of the pulse count/width preset register 1408, if in PWM mode, or is one-half the value stored in the cycle time preset register 1422, if in PTO mode.

If the pulse width preset value is 0, then **PLSxOUT** will be low for the entire cycle. If the pulse width preset value is equal or greater than the cycle time preset, then **PLSxOUT** will be high for the entire cycle.

The pulse count/width preset register 1408 is loaded by software but only the lower 16-bits are significant when operating in PWM mode. Therefore, software should only write the least significant word of the register in PWM mode. The valid range of this register in this mode is 0 to 65535. This register may be loaded by software only when the PLS block is disabled.

Cycle Quantity Control (PTO mode only) - A 32-bit up counter, called the *pulse counter 1428*, is driven on each *CycleDone* event to count the number of cycles generated on the **PLSxOUT** signal. The value stored in this counter is compared by comparator 1434 to the value in the 32-bit pulse count preset register 1416 to determine completion of the current PTO operation. When the pulse counter 1428 value reaches the pulse count preset value, the ***PTOComplete*** event is generated. In PWM mode, this counter is not used.

In PTO mode, the full 32-bits of the pulse count/width preset register 1416 is significant and is loaded by software. The valid range of this register in this

mode is 1 to (2^{32}-1). This register may be loaded by software only when the PLS block is disabled.

Operational Control - A control register contains bits to allow the software to enable or disable the operation of the PLS block. Once software enables the PLS block, the switch-over of the output point from **QBxPS** control to PLS block control as well as the operation of internal counters and comparisons is synchronized with the next rising edge of the time base clock. This is done to ensure that the initial cycle of the output wave-form is of proper duration and pulse width. This register is held at its reset state in the case of watch dog time out fault condition or due to direct software control of the output disable function.

A setup register contains bits to allow the software to configure the operation of the PLS block. Included are a PLS time base select and an operating mode select. An interrupt enable register gives the software the ability to individually enable and disable interrupt generation for the *PTOComplete* and *AdderError* interrupt events. A status register is provided to allow the software to obtain information about the current operational state of the PLS block. This information includes the enabled state and the state of the pipeline loaded flag (described below).

An interrupt status register provides information on the interrupt source or sources generating the interrupt condition. The **INTxPLS** signal originates from the PLS block and is used by the Interrupt Control Unit. This signal becomes active when one or both of the PLS interrupt-generating events has occurred (subject to the state of the interrupt enable bits in the interrupt enable register). Reading the interrupt status register will clear all active interrupt indications in the PLS block and release the **INTxPLS** signal. Also, disabling an interrupt generation source in the interrupt enable register will clear a pending interrupt from that source. If no other interrupts are then pending in the PLS block, the **INTxPLS** signal will be released.

Hardware Pipeline Mechanism - The pipeline mechanism is provided so that modifications to the wave-form characteristics can be applied with no resultant wave-form distortion. The following pipeline registers are provided to hold the new values for cycle time, pulse width, etc:

- Cycle time pipeline register is used to store the new cycle time value.
- Delta cycle time pipeline register is used to store the new delta cycle time value.
- Pulse count/width pipeline register, is used to store the new pulse count or pulse width value (if in PTO mode, this register stores a new pulse count and the full 32-bits of the register are used. If in PWM mode, this register stores a new pulse width value and only the lower 16-bits of the register are used).

A *pipeline loaded flag* is implemented as part of the interface between the PLS block and software when using the hardware pipeline mechanism. The pipeline loaded flag is set whenever the software writes a value to the least significant byte of the pulse count/width pipeline register. Therefore, to use the hardware pipeline mechanism, the software must always write a value into the pulse count/width pipeline register and it must be the last pipeline register to be written.

All pipeline registers must be loaded by software since the contents of all pipeline registers will be transferred to the operating registers. In PWM mode, software does not need to write to the delta cycle time pipeline register because the delta cycle time register is held at 0 while the PLS block is in PWM mode. Any contents of that pipeline register are effectively discarded.

Once the pipeline registers are written and the pipeline loaded flag is set, the contents of the pipeline registers will be transferred to the operating registers under the following conditions:

- If the PLS block is enabled, the transfer will be made upon the occurrence of the *PTOComplete* event, if in PTO mode, or the *CycleDone* event, if in PWM mode.

- If the PLS block is disabled, the transfer will be made immediately when the PLS block is enabled by software

After the transfer to the operating registers is made, the pipeline loaded flag will then be automatically cleared.

Once the pipeline loaded flag is set in an enabled PLS block, the software is prevented from writing into any of the pipeline registers until their contents have been transferred into the operating registers (signified by the clearing of the pipeline loaded flag by the ASIC). If the PLS block is disabled, then the pipeline registers may be written by software with no restrictions and each time that the least significant byte of the pulse count/width pipeline register is written, the pipeline loaded flag will be set.

Auto-disable Mechanism - An enabled PLS block in PTO mode can be automatically disabled under the following conditions:

- The *PTOComplete* event occurs and there is no new wave-form setup stored in the pipeline registers (the pipeline loaded flag is 0).

- The *AdderError* event occurs.

When the PLS block is automatically disabled, the master enable bit in the control register is cleared by the ASIC. The pipeline loaded flag is not affected.

Manual Disable - Whenever the master enable bit in the control register is cleared by software, the PLS block will immediately enter the PTO Disabled

State or the PWM Disabled State (described below), dependent upon the current operating mode of the block. A software disable will also clear the pipeline loaded flag and any pending interrupts.

Valid Register Ranges - The table in figure 14b shows the range of valid values for each of the numeric registers in the PLS block.

PTO Operation

This mode is selected by enabling the PLS block. Software will control the cycle time, the delta cycle time, and the pulse count. The duty cycle is fixed at 50%. Therefore, the following functionality of the PLS block is enabled for use in this mode:

- The cycle time preset and cycle time pipeline registers.
- The delta cycle time and delta cycle time pipeline registers.
- The pulse count/width preset and pulse count/width pipeline registers (configured to hold the 32-bit pulse count).
- The 32-bit pulse counter.
- The 16-bit cycle time counter.
- The comparison to generate the *CycleDone* event.
- The comparison to generate the *PTOComplete* event.
- The comparison to control the state of PLSxOUT.
- The pipeline loaded flag.
- The *PTOComplete* and the *AdderError* interrupts.

Major state transitions of the PLS block in PTO mode are shown in the state diagram 1500 of Figure 15.

When the PLS block is in the **PTO disabled state** 1502:

- The **QB0.x** output is controlled by Output Point State Register instead of the PLS block.
- The cycle time counter and the pulse counter are held at 0.
- The PLSxSTAT.EN status bit indicates disabled.

When the PLS block is in the **PTO operating state** 1504,

- The **QB0.x** output is controlled by the PLS block (through the PLSxOUT signal).
- The PLSxSTAT.EN status bit indicates enabled.
- Normal PTO operation is enabled.

The cycle time counter is incremented on the rising edge of the PLS time base and most of the functionality of the PLS block is driven, either directly or indirectly, from that counter. The segment in figure 16 of pseudo-code outlines the ASIC actions in PTO mode occurring on a rising edge of the PLS time base (the order of the actions is important, so assume top-to-bottom evaluation of these statements). The skilled practitioner will instantly appreciate the operation of the ASIC in PTO mode from the pseudo-code.

PWM Operation

This mode is selected by enabling the PLS block. Software will control the cycle time and the pulse width. Therefore, the following functionality of the PLS block is enabled for use in this mode:

- The cycle time preset and cycle time pipeline registers.
- The pulse count/width preset and pipeline registers (configured for a 16-bit pulse width).
- The 16-bit cycle time counter.
- The comparison to generate the *CycleDone* event.
- The comparison to control the state of PLSxOUT.

- The pipeline loaded flag.

The delta cycle time register and the delta cycle time pipeline register are held at 0 and the pulse counter and interrupt generation for the block are disabled. The INTxPLS signal should always be driven to the inactive state in this mode.

Major state transitions of the PLS block in PWM mode are shown in the state diagram shown in figure 17.

When the PLS block is in the **PWM disabled state** 1702:

- The **QB0.x** output is controlled by the Output Point State Register instead of the PLS block.
- The cycle time counter is held at 0.
- The PLSxSTAT.EN status bit indicates disabled.

When the PLS block is in the **PWM operating state** 1704,

- The **QB0.x** output is controlled by the PLS block (through the PLSxOUT signal).
- The PLSxSTAT.EN status bit indicates enabled.
- Normal PWM operation is enabled.

The cycle time counter is incremented on the rising edge of the PLS time base and most of the functionality of the PLS block is driven, either directly or indirectly, from that counter. The segment shown in figure 18 of pseudo-code outlines the ASIC actions in PWM mode occurring on a rising edge of the PLS time base (the order of the actions is important, so assume top-to-bottom evaluation of these statements). The skilled practitioner will instantly appreciate the operation of the ASIC in PWM mode from the pseudo-code.

Pulse Train Output Profile Instruction (PTOP)

The pulse output function provided by the ASIC is primarily intended for use in controlling stepper motors. The PLC provides a user instruction for the generation of profiles for acceleration/deceleration and running of stepper motors. This instruction is shown in figure 19.

Expansion I/O Unit

Overview of Expansion Bus Concept

Physically, the expansion subsystem consists of enclosed I/O modules which are mounted independently, without a common back plane. A ten conductor ribbon cable which is part of each module is used to connect to the preceding module or the CPU. This cable carries the 5V power and logic signals required for the modules operation. See figure 20. The signals in the I/O expansion bus are described in figure 21.

While all signals are physically daisy-chained through the modules, only EMA signals are logically broken and regenerated at each module. All other signals form a single electrical bus that extends the length of the I/O module chain. The value of the EMA signals is incremented before it is passed to the next module. Each access to the expansion I/O bus includes a module address field for selection of the module to be accessed. Only the module whose address matches the module address field of the message will respond to the access. This scheme allows arbitrary mixes of I/O module types to be concatenated without address switches and without a fixed back plane. Logical expansion of up to 8 units is possible with three EMA signal lines, however only 7 modules are allowed at present. Of course, other expansion units are possible.

Figure 22 shows the logical structure 2200 and the signal flow of I/O modules 2202 in the present invention. The details of the I/O expansion logical

structures are not herein described. The table in figure 23 gives the values for each of the components used in the electrical interface of an expansion I/O module. The tables in figure 24 describe the electrical characteristics of the I/O bus signals. Access to expansion I/O is controlled by the ASIC in the CPU and will be understood from the diagrams in figures 25 and 26 which illustrate the read and write cycle sequences as seen on the expansion I/O bus.

The ASIC (Master State Machine and Transaction Timing)

The following describes the bus transaction sequence. EM bus timing is arranged to provide time for propagation delay and signal skew where needed. The master provides signals on falling edge clocks of EMC, to be latched by the slave on the next rising edge. One half EMC clock time is allowed to cover set up of the data to the slave, and allow for worst case skew of EMC vs. EMD through the ASIC buffers, external cables, and possible repeater station buffers. The slave presents data on rising edges of EMC, and the master latches data on the following rising clock edge. This allows a complete clock time for propagation of the clock to the slave, slave clock-to-output delay, and return propagation time.

During a bus transaction, register fields are presented on the bus most significant bit first. The P1 bit of the 2-bit parity field is presented first, then the P0 bit. The master state machine actions are described with respect to both the rising and falling edges of EMC while the slave is assumed to have no other time base than EMC and may operate completely on EMC rising edges. EMC is driven low when not in use, to minimize possibility of clocking false data into slaves.

Bus Transaction Sequence

The following paragraphs describe the various components of the EM bus transaction sequence, both when writing data to the slave and when reading data from the slave. The following actions are assumed to be generated from

the master (the EM Unit) unless they are specifically attributed to be performed by the slave. The following text and the subsequent clocking diagram will assist in understanding the transaction sequence.

Sequence Start - An EM bus transaction is triggered by a software write to the EM0START or EM1START register. The in-progress bit of the EM status register is set immediately. Further write accesses to EM registers are disabled by the ASIC while the in-progress bit is set to avoid disturbing the bus sequence in progress. It is the responsibility of software to ensure that new bus transactions are triggered only when the EM Unit is idle (in-progress bit is low). The bus transaction is clocked from the EMC0 signal, if the sequence was triggered by a write to the EM0START register. If the sequence was triggered by a write to the EM1START register, then the bus transaction is clocked from the EMC1 signal. The actual start of the EM bus sequence (referred to as EMCST) occurs on the first falling edge of the EMC time base after the transaction trigger.

Module / Register Address - At point EMCST, the master drives the XA_OD signal low and enables the EMC and EMD signals. The detection of an active XA_OD on a rising edge of EMC is used by the slave to identify the start of an addressing operation. The master then produces 10 clocks on the EMC signal, clocking the contents of the EMADDR register and 2 parity bits, P1 and P0, in sequence on EMD. The first bit of EMADDR presented on EMD is valid at the EMCST point. The XA_OD signal is driven high again at EMC↓ 1. The master presents a new bit on EMD at each EMC↓. The EMD signal is released at EMC↓ 10. The slave latches valid information at EMC↑ 1:10 (rising edge of EMC 1:10) .

Read Data Sequence - For a read transaction, the EMDDIR signal is driven to the low state at EMC↓ 10. After the clocking of the module/register address bits, the master produces 13 more clocks (11:23) on the EMC signal. EMC clocks

11:12 are idle clocks to allow the slave time to make the appropriate register data and parity available for transfer. The selected slave provides 8 data and 2 parity bits on the EMD signal at EMC↑ 13:22 and returns to idle state at EMC↑ 23 (the slave releases the EMD signal at this time). The master latches information at EMC↑ 14:23. The slave provides the most significant bit of the data value first, then the P1 parity bit, and finally the P0 parity bit. The data value is stored in register EMDATA. The master determines the parity of the value stored in EMDATA and compares this calculated parity against the parity bits received from the slave. If the calculated parity and received parity do not agree, then the error bit in the EM status register is set. Otherwise, the error bit is cleared. The EMDDIR signal is released (driven high) on EMC↓ 23.

Write Data Sequence - For a write transaction, after the clocking of the module/register address bits, the master provides 14 more clocks (11:24) on EMC. The master provides the 8 data bits (from the EMDATA register) on EMD followed by the 2 parity bits (P1 then P0). The master presents the data and parity bits at EMC↓ 10:19, then releases the EMD signal and drives the EMDDIR signal to the low state on EMC↓ 20. The slave latches valid information at EMC↑ 11:20 . The slave responds to a complete write cycle and correct parity by clocking the bits **0** then **1** from EMD at EMC↑ 22:23. The master latches this acknowledge response on EMC↑ 23:24. The slave returns to idle state and releases EMD at EMC↑ 24. If the 2-bit acknowledge value read by the master is not **01**, then the error bit in the EM status register is set. Otherwise, the acknowledge value is correct and the error bit is cleared. The EMDDIR signal is released (driven high) at EMC↓ 24.

Sequence End - The end of the bus sequence is defined to occur on the first rising edge of the EMC time base following EMC↓ 23 (read transaction) or EMC↓

24 (write transaction). At EMCND, the in-progress bit of the EM status register is cleared. The error bit must be valid when the in-progress bit is cleared. The bus signals should be at their idle states.

Parity Check Bits

Two parity bits are used to indicate the parity of a single 8-bit value. Each of the 2 parity bits represents odd parity on 5 bits of the value, as shown in the table of figure 27. Odd parity on odd number of bits ensures that fields of all 1's or all 0's have opposite polarity parity bits.

Software Interface Considerations

A typical EM bus transaction will include software writing the EMADDR register, writing a data byte to the EMDATA register (if a write transaction), then writing to the EM0START or EM1START registers to initiate the sequence. The software will repeatedly read the in-progress bit of the EM status register to determine when the transaction is complete. On completion of the EM bus transaction, the in progress bit will be cleared and the error bit will be set appropriately. The data value can now be read from the EMDATA register if a read transaction was performed. Software may ensure that only the EMSTAT register is read while the transaction is in progress and the EM Unit may be designed to correctly handle a read access of that register by the software at any time.

If an interrupt event occurs while a EM bus transaction is in progress and the interrupting routine requires the use of the EM Unit, then that routine may sample the state of the in progress bit and wait for the transaction to complete before accessing any other EM Unit registers. Such interrupt routines will have an obligation to save and restore the contents of the EMADDR and EMDATA registers as well as the state of the error bit. This is required so that those registers will contain the correct values for the bus transaction started by the interrupted routine once control is returned back to that routine.

Output Disable using the XA_OD signal

The XA_OD signal serves the dual use of identifying the start of address cycle and the occurrence of a output disable condition (caused by a CPU system fault). At the slowest expected bus rate of ~ 1 MHz, the longest normal assertion of XA_OD in address usage is ~ 1 μs. A threshold of ~ 100 times this, or 100 μs for slave recognition of XA_OD as an output disable allows for a simple diode/R/C filter at the slave and gives an adequately responsive output disable function. The XA_OD external signal will be the logical OR of the address cycle identification signal from the EM bus state machine and the outputs OK signal originating from the Watchdog Timer Unit.

User Program Memory

User program memory may be used to store the user program. The end of the main program may be terminated by the use of a MEND (main program end) instruction. The remaining portion of user program memory beyond the MEND instruction may be reserved for subroutines, and interrupt routines.

Serial Communication

The PLC provides a serial communication port as described with reference to figure 1. The communication port uses RS485 signal levels and is capable of running at 9.6 KB, 19.2 KB or 187.5 KB, while supporting the system communication protocols (PPI or DP/T). This port serves as the programmer interface to the PLC. In addition the port is available in the run mode for use as a general purpose communication port (referred to as freeport mode) under the complete control of the user program. In free port mode the communication port is capable of operating at baud rates from 300 baud to 38.4KB. The physical connection to the RS485 port is provided by a nine pin D connector.

Special Memory (SM) bits (see section 6.2, SM2, SM3, SM30 and SM130) are provided for free port use of the communications port. The default state of the special memory bits that control the use of the communication port enables the use of the communication port as a programmer interface. If the user program turns on the special memory bit that controls the communication port's use, the user program will be enabled to send and receive messages through the communication port. In this mode of operation the user can configure the port for any character based protocol desired (7 or 8 bits/character, odd, even or no parity, and baud rate). The port will operate in half duplex mode and the user program can operate the port in polled or interrupt mode. A receive character buffer is provided in the SM user area, along with a communication status register, and a configuration register, which includes the bit that controls the port's mode of operation. All communication functions except transmitting can be performed through the special registers in SM memory. Transmitting can only be accomplished using the special write port message instruction. The user program never has direct access to the communications controller or the interrupt structure of the machine.

PPI Master and Slave Modes Network Read and Write

The PLC may function as a PPI Slave. In PPI Slave mode the PLC acts as a responder to communication messages.

In one arrangement of the invention, when the CPU is placed in the Run mode, the PLC can function as a PPI Master as well as a PPI Slave. The user can select PPI Master mode under the control of the user program stored in user program memory. In this arrangement, a well-known token-ring communication protocol may be implemented to effect communication between master and slave. In PPI Master mode the PLC is capable of holding the token and initiating PPI requests thereby enabling the user to access data in other CPUs via the

network read and write instructions. When the CPU is not holding the token, it responds to all requests as a PPI slave. The communication buffer sizes for PPI communication are defined in the table shown in figure 28.

While the PLC is functioning as a PPI Master, it will maintain the network in accord with well-known communication standards (such as the PROFIBUS definition provided by the EN 50170 European Standard). The PLC will use the following algorithm for calculating the target token rotation time.

$$TTTR = (8 + HSA)(256)(1/BR)$$

where

TTTR - is the target token rotation time
HSA - is the highest station address
BR - is the baud rate in bits per second

If upon receipt of the token, the target token rotation timer has expired, the PLC will immediately pass the token to the next token holder without sending any messages that may have been queued prior to receipt of the token.

Freeport Link Provided in PPI Master Mode

The present invention provides free port operation of the communications port, hereinafter "freeport", that allows the user to control the port either manually or by operation of a user program. The procedure of freeport operation is shown in figure 30a.

In the instant invention, the user implements freeport (step S3000) use by forcing an interrupt (step S3002) of the PLC according to the afore-described interrupt architecture. In this case, the user initiates freeport communication while operating the PLC in master mode by attaching an interrupt event that will pass control of the communications port to the user program (step S3004). This is performed while the system protocol holds the token (step S3006).

In response, the PLC passes control to the user program (step S3008). The PLC controls the UART settings (step S3010) in accordance with the SM bit definitions SMB30 and SMB130 shown in figure 30b. In that instance, the mode bits of SMB30 and SMB130 definitions are set by the PLC to PPI master mode (step S3012).

Once control is passed from the system to the user's interrupt routine, the user program controls transmission and reception of messages using the freeport (step S3014). Once the user program has completed its task, the user program may pass control back to the system by the execution of a special instruction that terminates freeport operation (step S3016). The instruction used to terminate freeport operation is described in the table shown in figure 29. The system will then resume normal operation by passing the token to the next station (step S3018). The system is responsible for maintaining all aspects of the network just as if the user program had never been given a time slice during the token hold time of the PLC (step S3020).

Built In Freeport Protocol

In order to provide higher performance for communication using PPI protocol, a built-in protocol selection option is provided for the PPI protocol. When the PLC is placed in the RUN mode and the freeport PPI protocol is selected by the value written to SM30 or SM130, the communication port will be configured according to the values written into SM30 or SM130. When freeport mode is exited, the PLC will revert back to the default PPI protocol at the selected baud rate. The code for selecting freeport PPI protocol will be as defined by SM30 (port 0) and by SM130 (port 1). The table in figure 30b gives the definition of SM30 and SM130.

Freeport Receive Message Interrupt

In order to simplify the programming for freeport communications, the PLC supports a receive message instruction that may be used in conjunction with a receive message event interrupt. The user specifies the conditions that are required to define the beginning and end of a message. The PLC supports the following message condition specifications:

Port 0:

SMB86 - Message Status Byte: Indicates termination conditions

SMB87 - Message Control Byte: Enables the selected message receive criteria

SMB88 - Start Character: The character that uniquely identifies the start of a new message

SMB89 - End Character: The character that uniquely identifies the end of the message

SMW90 - Idle Line: Time period after which the first character received starts a new message

SMW92 - Inter-character/Message Time Out: The maximum time allowed between characters or for the message to be received. Usage is determined by the message control byte.

SMB94 - Character Count: The maximum length of the receive message

Port 1:

SMB86 - Message Status Byte: Indicates termination conditions

SMB87 - Message Control Byte: Enables the selected message receive criteria

SMB88 - Start Character: The character that uniquely identifies the start of a new message

SMB89 - End Character: The character that uniquely identifies the end of the message

SMW90 - Idle Line: Time period after which the first character received starts a new message

SMW92 - Inter-character/Message Time Out: The maximum time allowed between characters or for the message to be received. Usage is determined by the message control byte.

SMB94 - Character Count: The maximum length of the receive message

The first byte of the receive buffer is used as a character counter which may indicate how many characters were received by the receive message function. The definition of each of these special memory registers is given in the table shown in figures 30c and 30d.

DP_S7 Communications

The PLC may support the DP/T protocol that provides a MPI transport for Master-Slave communications. PPI messages are completely compatible with the DP/T messages, so no special selection of protocol is required. The PLC may accept either a PPI request or a DP/T request and respond accordingly. Support for DP/T communications over the PPI port may include being HD4 compliant on transmissions from the PLC at all baud rates supported.

While PPI is a connectionless protocol, DP/T is a connection oriented protocol. The table of figure 30d defines the number of DP/T connections and the buffer size for each connection supported by each PLC on each port. This table does not include the buffer for the default SAP that may be supported on each port.

Modem Communications

The CPU of the present invention supports communications over standard 10-bit, full duplex modems. This is considerably important because the newer 11-bit modem in wide useage is often times too expensive. It was discovered that this is particularly true in the industry setting where industrial technology typically lags behind the state-of-the-art in computer hardware. On the other hand, the old 10-

bit modems are plenty to be found on the industrial floor. This invention takes advantage of the discovery that there are a surplus of 10 bit modems in industry and, therefore, provides a unique 10-bit protocol that allows the PLC of the present invention (or any PLC or processor for that matter) to connect to factory floors so equipped. Of course, the 10-bit protocol of the present invention is valuable, and is applicable, to other settings where 10-bit modems are employed.

Another advantage of the 10-bit protocol of the present invention is that the 10-bit protocol is compatible with higher order bit protocols. Thus, the 10-bit protocol may be received by an 11-bit modem or higher.

The 10-bit protocol of the present invention is counter to conventional wisdom. Of course, it runs counter to intuition to employ lesser bits than is capable by the latest modem technology. It is even further contrary to conventional wisdom to employ the 10-bit modem protocol because the present invention sacrifices the parity bit, the linch-pin upon which all remote communications rely to ensure integrity of data. Indeed, when it is considered that PLCs are responsible for controlling machinery, e.g., heavy equipment or precision robotics, it is realized that the company may be put out of business or, worse, someone is injured by the improper commands of a PLC. The ordinary practitioner would counsel strongly against a 10-bit protocol.

The 10-bit protocol of the present invention is described with reference to figure 31. In one arrangement, the protocol used for modem communications is a modification of the standard PPI protocol. Of course, other bit assignments are possible. The PPM or Point to Point Modem protocol specified here assigns the following bits:

1 - Start bit
8 - Data bits

1 - Stop bit

0 - Parity bits

In the PPM protocol each request from the master receives a response without an intervening acknowledgment or poll (step S3100). It is recommended that only SD2 message types defined for PPI protocol are used in the PPM protocol (step S3104). In order to fulfill this requirement the master device may specify the use of the Master/Slave protocol (DP/T) in the establish association request (step S3106). After each request, the master may allow, for example, 15 seconds for the slave device to respond before timing out the transaction (step S3108). Upon receipt of a request, the slave will allow a minimum turn around time of, for example, 20 milliseconds before issuing the response (step S3110).

Each request/response message is preferably a SD2 PPI message (step S3112) with a sixteen bit CRC check code appended as the last two bytes of the message (step S3114). The CRC uses the industry standard polynomial (X16 + X12 + X5 + 1) which is known as the CRC-CCITT polynomial (step S3116). The CRC is calculated over the entire SD2 frame excluding start and stop bits (step S3118). The CRC generator may be preset to all ones for both transmission and reception (step S3120). In this case, receipt of a valid message produces a pattern of 0x1D0F (step S3122).

The PPM protocol is signaled to the PLC by grounding pin 9 of the communication port connector. The PLC provides an internal pullup resistor on this pin in order to select standard PROFIBUS protocols.

It will be appreciated that the 10-bit modem protocol of the present invention is not necessarily limited to 10 bits and may be extended to any number of bit configurations including, for example, {1,2,3...9, 10, 11 ... ∞}. Indeed, the present invention may be better considered an n-bit protocol that does not

include a parity bit. The n-bit protocol may be implemented using the foregoing description to ensure data integrity.

Hiding Portions of the User Program

The PLC of the present invention provides for the protection of proprietary software or code. The same feature allows the PLC to hide portions of code from the user that may not be intended for the user's viewing such as code generated by the PLC for performing routine operations of the PLC. It will be appreciated that this function encourages third party software developers to provide software who would otherwise shy away because the code is otherwise available to users.

To that end, the PLC supports a HIDE instruction that allows the user (or programming device) to identify and hide portions of the user program. The PLC programs are transferred and stored in the user program memory with the hidden code sections encrypted. The PLC decrypts the hidden code sections at compile time for the purpose of generating the executable program. Once compilation has been completed, the hidden code sections are re-encrypted.

A user supplied password capability is provided that is used to encrypt the portion of the program that is to be hidden. In this way only the person with authorization will be able to view the program sections that are hidden.

As shown in figure 32a, the hide instruction of the present invention marks the start of the encrypted portions. In the invention, a flag P is set to indicate that encrypted code exists. Now with reference to figure 32b, the operation of the hide function shall be described. In Step S3200, the user invokes the hide instruction for a particular piece of proprietary code. In response, the PLC causes the P flag to be set (Step S3202). The PLC encrypts the code with a password that may be provided by the user (Step S3204) and inserts the hide

label in the user memory at the beginning and end of the code to ear-mark the encrypted portion (Step S3206). Upon compilation of the user program (Step S3208), the PLC causes the encrypted portion to be decrypted (Step S3210) and the user program executes normally (Step S3212). Once execution of the program is complete, the PLC re-encrypts the proprietary code (Step S3214).

User Passwords

Restricted access to the PLC functions and memory may be provided through the use of a password which allows the user to configure access restrictions. Without a password the PLC may provide unrestricted access. When the PLC is password protected, the PLC may prohibit all operations that are restricted according to the configuration provided at the time the password was installed. The PLC may allow one station to legitimize the communication link so that the user of that station can have unrestricted access. The user who knows the password may reconfigure the restrictions at any time by entering the correct password and editing the restriction class. Users who do not know the password must live with the restrictions assigned by the person who knows the password.

In the event that the user forgets a password, the master password may be used to gain access to the PLC. For the master password to work, the PLC mode switch may be in either the STOP or TERM position. When the master password is used the following actions are taken by the PLC software:

1. The PLC will transition to the STOP mode
2. The user program (OB1) is cleared (deleted)
3. The user data memory (DB1) is cleared (deleted)
4. If SDB0 exists, all parameters of SDB2 except the station address and baud rate will be copied to SDB0
5. All SM flags are set to their default state

6. All flags are cleared
7. All SCR bits are cleared
8. Analog outputs will be frozen
9. All system data memory is set to the default state
10. All forced points are cleared and unforced
11. The time of day clock will not be changed
12. All timer/counter current data is cleared
13. The internal EEPROM is initialized (OB1 and DB1 are deleted and SDB2 is copied to SDB0 with the exception of the station address, and all inputs, outputs and data values are unforced)

System Function Call Support

The PLC 100 supports System Function Calls (SFCs) which allow the user to upload customized functions to the PLC.

In general, SFCs may be created, for example by the Siemens S7-200 development groups, and downloaded to the PLC by the programming device. The PLC supports at least two SFCs. The maximum size of each SFC is dependent on the resources available in the specific PLC. For the CPU envisioned in this invention the maximum size of each SFC is 8KB including block header information. SFCs do not necessarily require any user data, although allocation of system data may be provided. SFCs downloaded to the PLC become part of the operating system of that PLC. The PLC further allows the SFCs to be uploaded, downloaded, deleted and copied to the memory cartridge. The PLC reports the existence of SFCs in the appropriate directory communication functions.

The exemplary SFC instruction is shown in the table of figure 33a. Now in more detail, figure 33b shows the method of the system function call. In Step S3300,

the user alerts the PLC that an SFC download is desired. The PLC performs the necessary handshaking protocol for the download (Step S3302) and the SFC is downloaded (Step S3304). The PLC stores the downloaded function as part of the library of functions of the operating system (Step S3306).

Load New Operating System

The PLCs may use FLASH EPROM for the operating system storage support loading of a new operating system through the communication port. This is accomplished using a boot block in the FLASH EPROM that cannot be erased while installed in the PLC.

Loading of a new operating system can be accomplished by directly connecting a personal computer to the PLC using a PC/PPI cable. Use of PROFIBUS cards installed in a personal computer or part of a PG are not preferred because of the complexity of the PROFIBUS protocol which is the only protocol that can be used with these devices.

The definition of the protocol used to load the new operating system will be identified by design and each communication message will provide data integrity checks in the form of checksums or CRC codes. Verification of integrity of the new operating system is before normal operation is invoked, but after the FLASH EPROM has been programmed.

STL Status

It has been found that the conventional debugging system for debugging programs is flawed because the prior debugging system of the PLC debugs after execution of the entire program or significantly interferes with the timing of the

program execution. To that end, the PLC supports the acquisition of the results of executing a group of instructions at the instance each instruction in the group is executed. This novel feature is done in real time and while the PLC is executing the instruction. In order to implement this function, the PLC instruments the compiled code segment for which status is to be gathered.

In one arrangement, the STL status is implemented on a display, such as provided by a human-machine interface coupled to the PLC as well-known. The size of the status window on the display will be determined by the number of operand values that may be returned. The PLC may reserve, for example, a 200 byte buffer for operand values. The size of this buffer is based upon the amount of data that can be returned in a single communication frame.

Only one STL status window is preferably supported. Supporting multiple windows requires additional RAM and it presents synchronization and triggering problems. Of course, with future developments in memory technology, it is possible that multiple windows are a possibility.

Only one device should be able to open an STL status window. The STL status window function will be associated with the station address of the device which opened the STL status window. As long as the STL status window remains open, all other requests for opening a STL status window will be rejected by the PLC

In order to open a STL status window the programming device may identify the address of the first instruction in the window and then identify the number of instructions in the window. The basic format of the window specification is shown below.

Instruction Address of Window Start	2 Bytes
No. of Instructions in STL Status Window	2 Bytes

In this example, the only instruction address specified is the memory address of the first instruction in the window. The programming device bears the complete burden of making sure that the response data requested in the setup of the STL status window fits in the 200 bytes of the response buffer.

It is a tenet that the STL status display must not cross the boundary between program parts, e.g., Main to Sub, Sub to Sub, Sub to Int, Int to Int. That would cause a disruption.

The exemplary format of the response buffer is shown in figure 34a, wherein the instruction address of the start of the STL status window is returned for positive confirmation that the PLC and the programming device are viewing the same window. The length field indicates how many bytes of data values are contained in the data area for each instruction. Now the codes will be described:

V 0 - data for this instruction is not valid because the instruction was not executed this scan.

1 - data for this instruction is valid.

ENO The enable output bit that indicates if the instruction executed without error

SCR The bit value of the SCR stack.

S4 - S0 S0 refers to the top of stack value and S1 refers to the next to top of stack value.

Data The STL status information returned for the instruction. The programming device is responsible for parsing the data based upon the request that set up the STL status window.

The PLC is responsible for making sure that the data captured in the STL status window comes from a single scan cycle. The PLC indicates this to the programming device using the Valid flag (V) that is provided with push down stack values for each instruction. Internally, the PLC tracks what is valid by using a scan counter value that is reset by a request for the STL status buffer and which is incremented each scan. When a value is to be written to the STL status buffer, the scan count status word for the buffer is compared to the current value of the scan counter. If the values are different, the scan counter value will be copied to the scan count status word for the STL status buffer and all V flags in the buffer will be cleared. Then the new values are stored in the buffer and the V flag for that instruction will be set. If the values of the scan counts are the same, then the new values will be stored in the buffer and the V flag for that instruction will be set. This allows the buffer to always be updated with the data from the last scan while keeping the data synchronized to the same scan.

Instrumenting the Compiled Code Window for STL Status

The apparatus and operation of the STL shall now be described simultaneously with reference to Figures 34b and 34c. The section of compiled code corresponding to the instructions in the STL Status Window is identified by the PLC (Step S3400) (shaded in the figure). The original compiled code in this shaded region is saved (Step S3402) and then restored once the STL Status operation is complete (S3420). Once the original compiled code has been saved, the instrumented code is compiled in another section of RAM (S3404).

In order to discuss the process of instrumenting the code a few terms need to be defined as follows:

Compiled instruction - A compiled instruction is the machine code or executable code created by compiling a STL instruction. Generally, each STL instruction is compiled into multiple machine instructions.

Instrumented instruction - An instrumented instruction is a compiled instruction along with the extra machine code required to save the status of power flow and the operand values.

In order to preserve the addresses of labels, for-next loops, subroutines and interrupt routines, each compiled instruction in the STL Status Window is replaced with a branch to the instrumented instruction (Step S3406). Each instrumented instruction is terminated with a return instruction which transfers control to the next compiled instruction (Step S3408).

The branch to the instrumented code is made using an LCALL instruction (Step S3410). The LCALL transfers control to the instrumented code that captures the STL status for that instruction (S3412). For those compiled instructions which are longer than three bytes, the return value is adjusted so that control will be transferred to the next compiled instruction (Step S3414).

The beginning address of the first compiled instruction in the STL Status Window may be determined before instrumentation (S3416). In order to reduce the amount of time required to determine this address, a table containing a pointer to the address of, for example, every twenty-fifth compiled instruction can be maintained (S3418). Such a table would require about 300 bytes for a 8192 byte user program. If more RAM is available, then the table of address pointers could point to every tenth compiled instruction which would further reduce the time for STL status instrumentation.

The PLC utilizes the table of figure 34c that lists the fastest Boolean instructions and shows what changes are required to compile the code so that all instructions

are a minimum of three bytes in length. The instructions in the shaded portions of the table tells the PLC which instructions which may be modified.

The foregoing technique requires the least amount of RAM, but it also requires that all compiled code instructions take at least the number of bytes required by the LCALL instruction that will be used to branch to the instrumented code section. For purposes of clarity it is assumed in the above that the LCALL instruction takes three bytes and that all compiled instructions take three or more bytes.

Design Concept

The mechanical design concept for the PLC family is shown in figure 1. The design concept is a "Brick" style that will have the ability to have units connected for expansion. These expansions units will also be of the same style and shape. The design goal is to make this unit as cost effective as possible while maintaining the same family appearance as the Siemens S7-300™ and S7-400 ™. The PLC family will have the same color and text style as the S7-300 ™ and S7-400 ™ family.

Housing

The PLC housing is molded in a uniform plastic and color (Noryl GFN1 SE1 Anthrazit 614 Herst N.R. GE 93263). The unit is made up of a top, a base, two access covers, one DIN rail latch, and two sizes of light pipes. All these parts snap together to allow for ease of manufacturing. Provisions in the housing is made to allow the PWB's to also snap into position. The I/O housing is designed in the same manner as the PLC housing. The housing will be designed for IP20 protection and to prevent other objects (such as a coin) from entering the unit. The table in figure 35a shows one possible set of dimensions of the units

Unit Mounting

The PLC 3900 shown in figure 35b is designed to be mounted either on a DIN rail 3906 or to be panel mounted 3908. A mounting latch will be used to lock the unit onto a standard type 50022 DIN rail when the unit is to be DIN rail mounted. When the unit is to be panel mounted, mounting holes are provided that accept either a metric M4 or a #8 type screw size. The PLC 3904 is shown here in engagement with the I/O expansion module 3902. A clearance of, for example, 25 mm will be required above and below the unit to allow the user wiring access and to allow for proper cooling of the unit.

While the present invention has been described with reference to specific values and circuit arrangements, it shall be appreciated that other values and circuit arrangements may be substituted for those disclosed which is within the spirit and scope of the present invention. In addition, it will be appreciated that the practitioner will instantly recognize how to implement the software code of the PLC and for that matter will understand that the various software applications may be written in any suitable programming language as well as stored in any suitable recording medium such as non-volatile memory include disc, CD-ROM or DVD, for example.

WE CLAIM:

1. An apparatus for a programmable logic controller (PLC) including an input filter for minimizing a number of filter elements by providing a circuit response simulating a capacitor driven by a constant current source whose output voltage is sensed by a comparator with a relatively large amount of hysteresis, said input filter comprising:

an input for receiving an input signal to be filtered; and

a circuit that applies to said input signal a circuit response simulating a capacitor driven by a constant current source whose output voltage is sensed by a comparator with a relatively large hysteresis to thereby minimize said number of filter elements of said input filter.

2. The apparatus of claim 1, wherein said input filter provides settable filter settings that control a filter function.

3. The apparatus of claim 2, wherein said PLC sets the filter settings according to a program executed by the PLC.

4. The apparatus of claim 1, wherein said circuit includes an up/down counter whose counting direction is controlled by a state of said input signal.

5. The apparatus of claim 4, wherein delay times of said input filter are set in accordance with a factor of a period of an input clock of said up/down counter.

6. The apparatus of claim 4, wherein said up/down counter counts until a predetermined number to detect an input transition of said input signal.

7. The apparatus of claim 6, wherein, when said predetermined number is counted by said up/down counter, said input filter causes an output signal of said circuit to be output and is caused to remain in an output state until a predetermined threshold is reached.

8. The apparatus of claim 1, wherein further comprising a clock for driving said input filter independently of a system clock of said PLC.

9. An apparatus for a programmable logic controller (PLC) including an input filter for minimizing a number of filter elements by providing a circuit response simulating a capacitor driven by a constant current source whose output voltage is sensed by a comparator with a relatively large amount of hysteresis, said input filter comprising:

input means for receiving an input signal to be filtered; and

circuit means that applies to said input signal a circuit response simulating a capacitor driven by a constant current source whose output voltage is sensed

by a comparator with a relatively large hysteresis to thereby minimize said number of filter elements of said input filter.

10. The apparatus of claim 9, wherein said input filter provides means for setting settable filter settings that control a filter function.

11. The apparatus of claim 2, wherein said PLC sets the filter settings according to a program executed by the PLC.

12. The apparatus of claim 9, wherein said circuit means includes up/down counter means for counting direction controlled by a state of said input signal.

13. The apparatus of claim 12, wherein delay times of said input filter are set in accordance with a factor of a period of an input clock of said up/down counter.

14. The apparatus of claim 12, wherein said up/down counter means counts until a predetermined number to detect an input transition of said input signal.

15. A method for filtering an input signal to a programmable logic controller (PLC) by minimizing a number of filter elements by providing a circuit

response simulating a capacitor driven by a constant current source whose output voltage is sensed by a comparator with a relatively large amount of hysteresis, said method comprising the steps of:

receiving an input signal to be filtered; and

applying to said input signal a circuit response simulating a capacitor driven by a constant current source whose output voltage is sensed by a comparator with a relatively large hysteresis to thereby minimize said number of filter elements of said input filter.

16. The method of claim 15, further comprising the step of setting settable filter settings that control a filter function of said input filter.

17. The method of claim 16, wherein the step of setting sets the filter settings according to a program executed by the PLC.

18. The method of claim 15, further comprising the step of incrementing/decrementing an up/down counter whose counting direction is controlled by a state of said input signal.

19. The method of claim 18, further comprising the step of setting delay times of said input filter in accordance with a factor of a period of an input clock of said up/down counter.

20. The method of claim 18, wherein the step of incrementing/decrementing said up/down counter counts until a predetermined number to detect an input transition of said input signal.

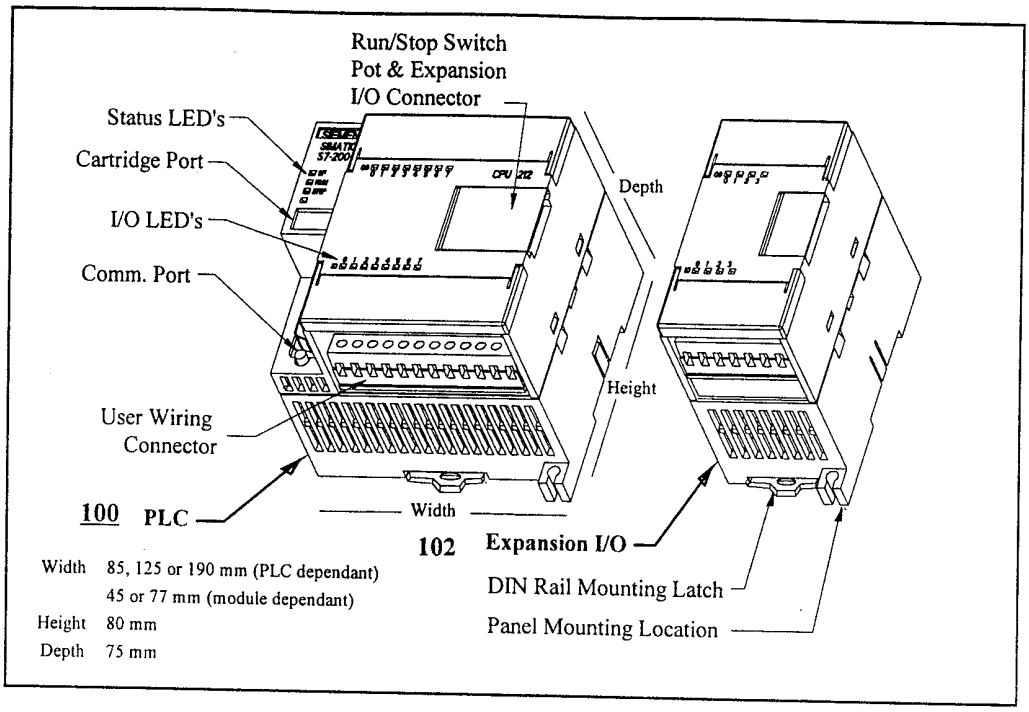


Figure 1

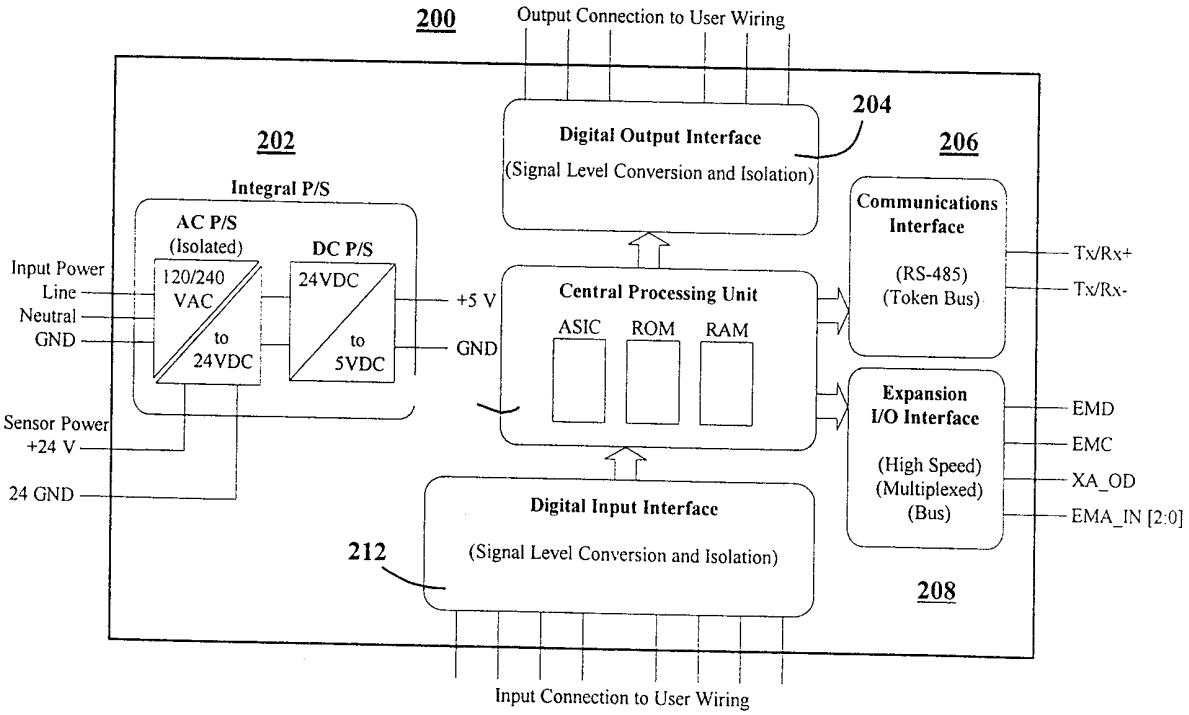


Figure 2

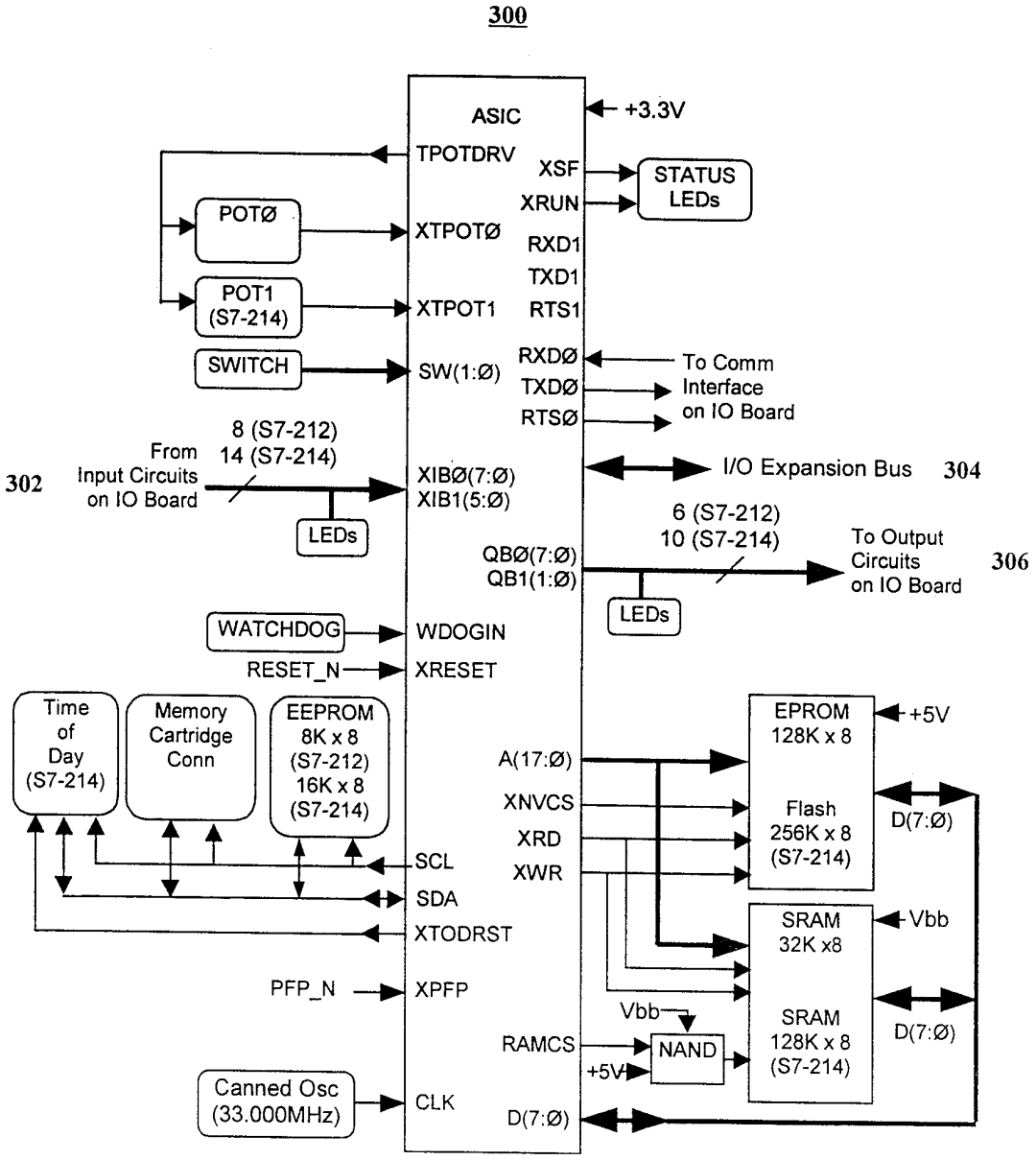

300
ASIC
+3.3V
TPOTDRV
XSF
STATUS LEDs
XRUN
POTØ
XTPOTØ
RXD1
TXD1
POT1 (S7-214)
XTPOT1
RTS1
SWITCH
SW(1:Ø)
RXDØ
To Comm Interface on IO Board
TXDØ
RTSØ
8 (S7-212)
14 (S7-214)
From Input Circuits on IO Board
302
XIBØ(7:Ø)
XIB1(5:Ø)
I/O Expansion Bus
304
LEDs
6 (S7-212)
10 (S7-214)
QBØ(7:Ø)
QB1(1:Ø)
To Output Circuits on IO Board
306
LEDs
WATCHDOG
WDOGIN
RESET_N
XRESET
Time of Day (S7-214)
Memory Cartridge Conn
EEPROM 8K x 8 (S7-212) 16K x 8 (S7-214)
A(17:Ø)
XNVCS
XRD
XWR
SCL
SDA
XTODRST
EPROM 128K x 8
+5V
Flash 256K x 8 (S7-214)
D(7:Ø)
SRAM 32K x8
Vbb
SRAM 128K x 8 (S7-214)
D(7:Ø)
PFP_N
XPFP
Vbb
RAMCS
NAND
+5V
Canned Osc (33.000MHz)
CLK
D(7:Ø)

Figure 3

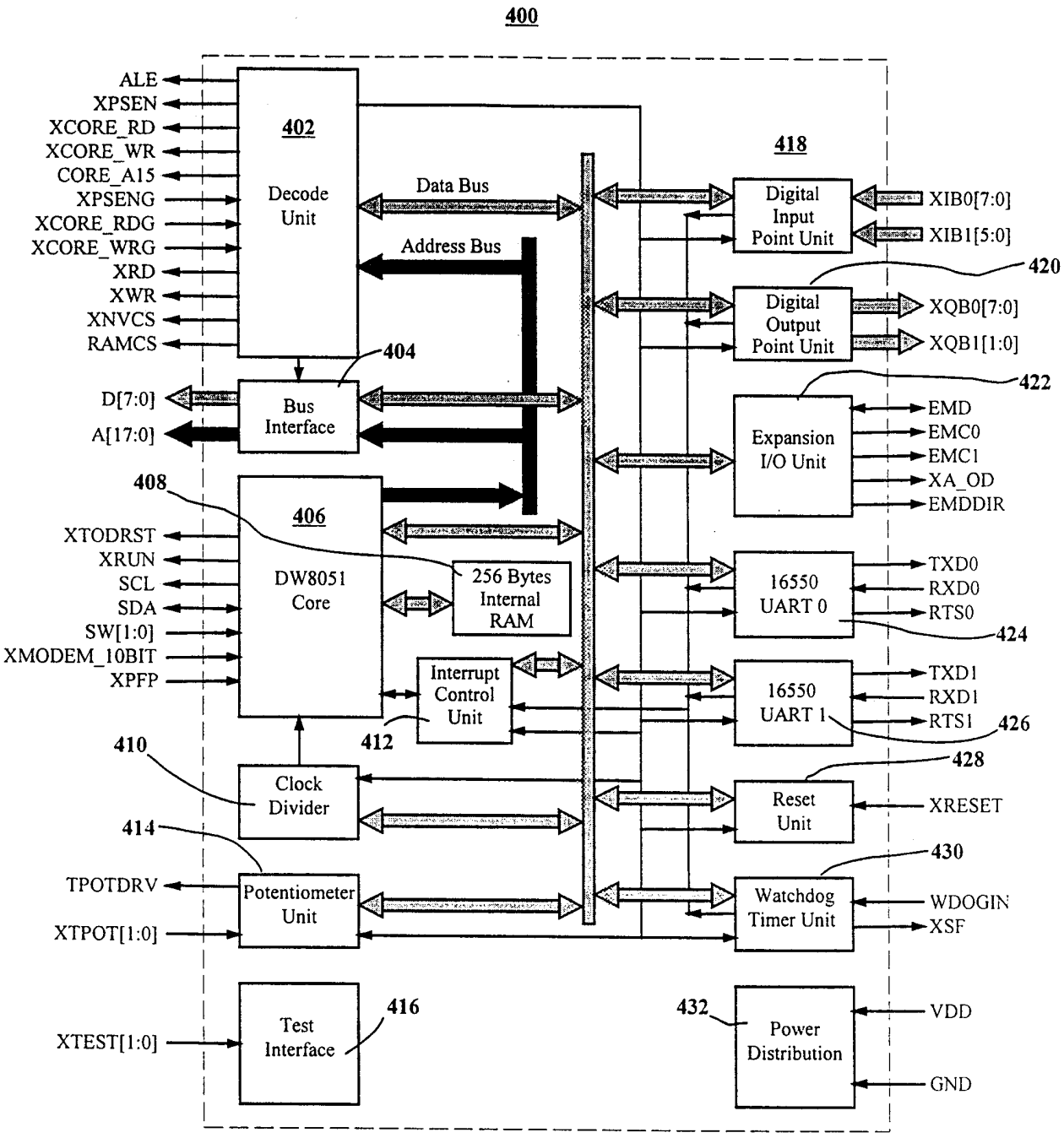

400
ALE
XPSEN
XCORE_RD
XCORE_WR
CORE_A15
XPSENG
XCORE_RDG
XCORE_WRG
XRD
XWR
XNVCS
RAMCS
402
Decode Unit
Data Bus
Address Bus
418
Digital Input Point Unit
XIB0[7:0]
XIB1[5:0]
420
Digital Output Point Unit
XQB0[7:0]
XQB1[1:0]
404
D[7:0]
A[17:0]
Bus Interface
422
Expansion I/O Unit
EMD
EMC0
EMC1
XA_OD
EMDDIR
408
406
XTODRST
XRUN
SCL
SDA
SW[1:0]
XMODEM_10BIT
XPFP
DW8051 Core
256 Bytes Internal RAM
16550 UART 0
TXD0
RXD0
RTS0
424
Interrupt Control Unit
412
16550 UART 1
TXD1
RXD1
RTS1
426
410
Clock Divider
428
Reset Unit
XRESET
414
TPOTDRV
XTPOT[1:0]
Potentiometer Unit
430
Watchdog Timer Unit
WDOGIN
XSF
XTEST[1:0]
Test Interface
416
432
Power Distribution
VDD
GND

Figure 4

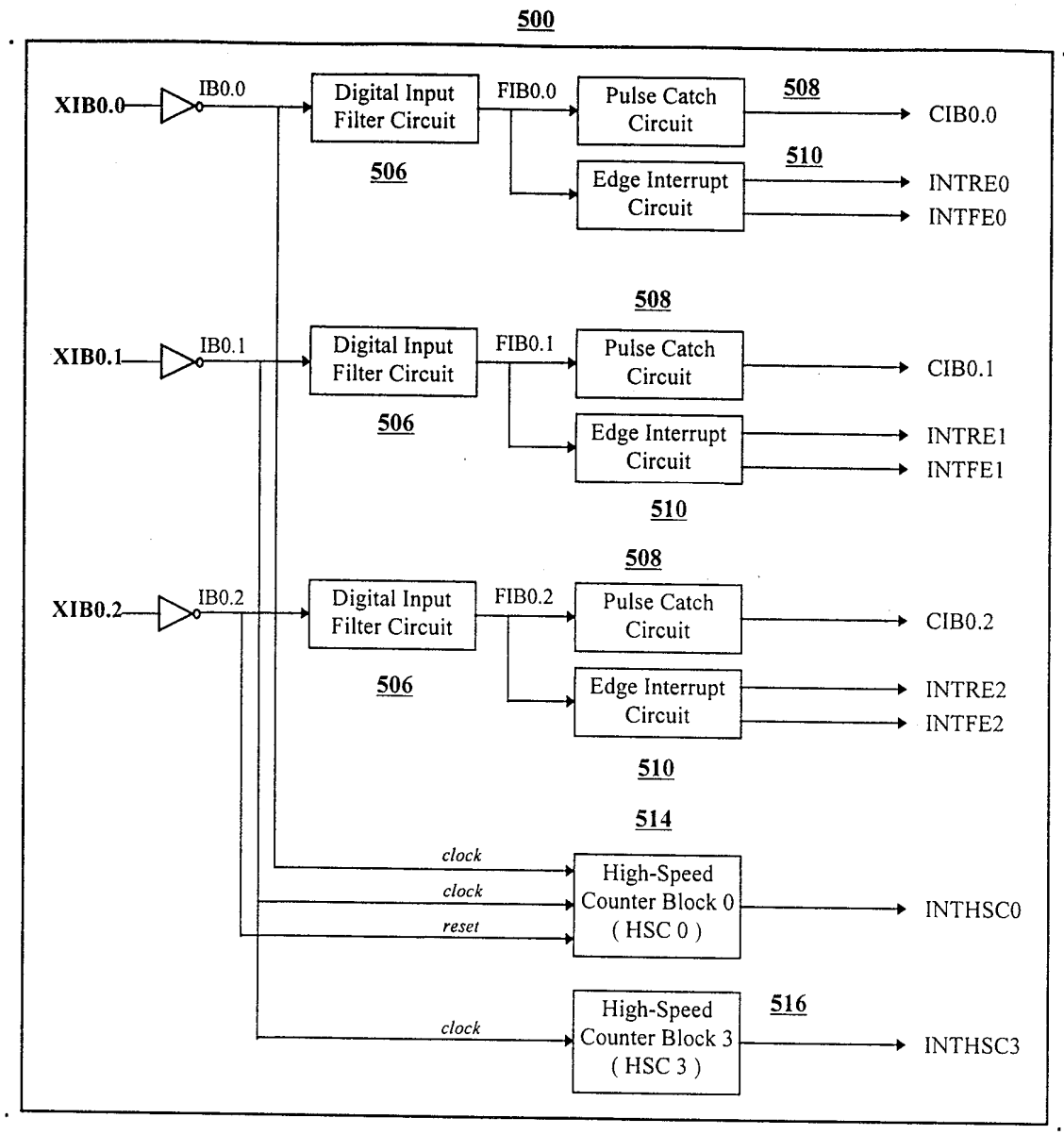

500
XIB0.0
IB0.0
Digital Input Filter Circuit
506
FIB0.0
Pulse Catch Circuit
508
CIB0.0
Edge Interrupt Circuit
510
INTRE0
INTFE0
XIB0.1
IB0.1
Digital Input Filter Circuit
506
FIB0.1
508
Pulse Catch Circuit
CIB0.1
Edge Interrupt Circuit
510
INTRE1
INTFE1
XIB0.2
IB0.2
Digital Input Filter Circuit
506
FIB0.2
508
Pulse Catch Circuit
CIB0.2
Edge Interrupt Circuit
510
INTRE2
INTFE2
514
clock
clock
reset
High-Speed Counter Block 0 (HSC 0)
INTHSC0
clock
High-Speed Counter Block 3 (HSC 3)
516
INTHSC3

Figure 5

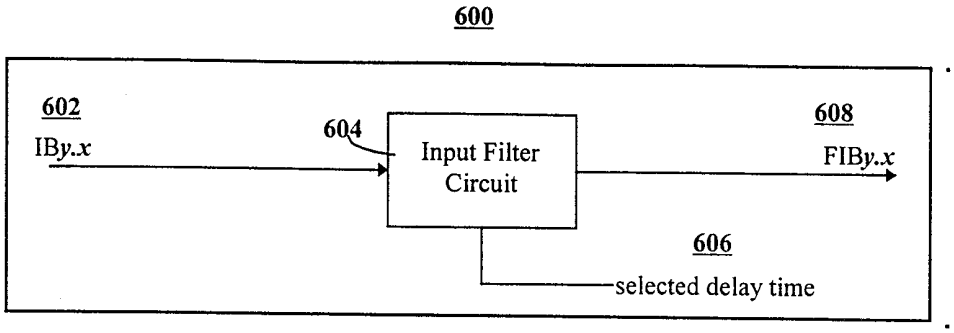


Figure 6a

Digital Filter Operation Definition

Input Point State	Current Count	Next Count	Present Output Value	Next Output Value
0 (decrements counter)	0	0	0	0
	n, where 3 ≥ n > 0	n - 1	0	0
	4	3	1	0
	n, where 15 ≥ n > 4	n - 1	1	1
1 (increments counter)	n, where 11 > n ≥ 0	n + 1	0	0
	11	12	0	1
	n, where 14 ≥ n > 11	n + 1	1	1
	15	15	1	1

Figure 6b

604

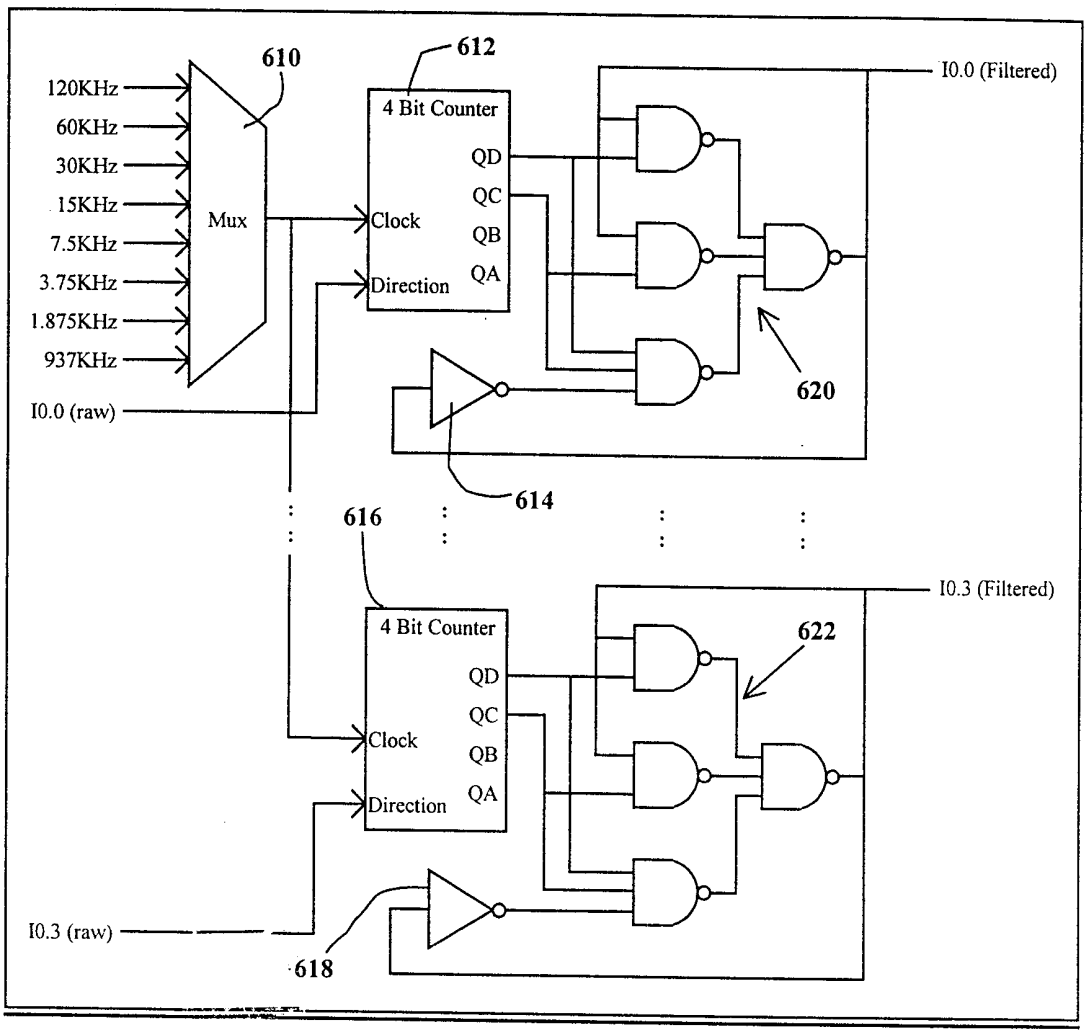


Figure 6c

Frequency	Period	Number of Counts	Delay time
120 KHz	8.33 μsec	12	0.1 ms
60 KHz	16.6 μsec	12	0.2 ms
30 KHz	33.3 μsec	12	0.4 ms
15 KHz	66.7 μsec	12	0.8 ms
7.5 KHz	133 μsec	12	1.6 ms
3.75 KHz	267 μsec	12	3.2 ms
1.875 KHz	533 μsec	12	6.4 ms
937 KHz	1067 μsec	12	12.8 ms

Figure 6d

Register Value	Corresponding delay time
00	0.2 ms
01	0.4 ms
02	0.8 ms
03	1.6 ms [1]
04	1.6 ms [1]
05	3.2 ms
06	6.4 ms
07	12.8 ms
08 to FF	no delay

[1] Two selections for 1.6 ms delay time exist for 1st generation ASIC compatibility reasons

Figure 7

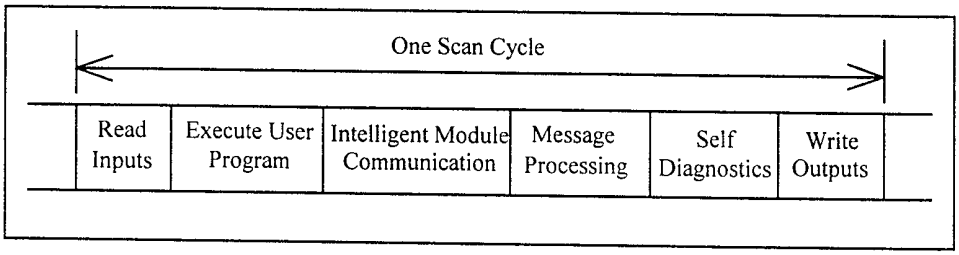


Figure 8

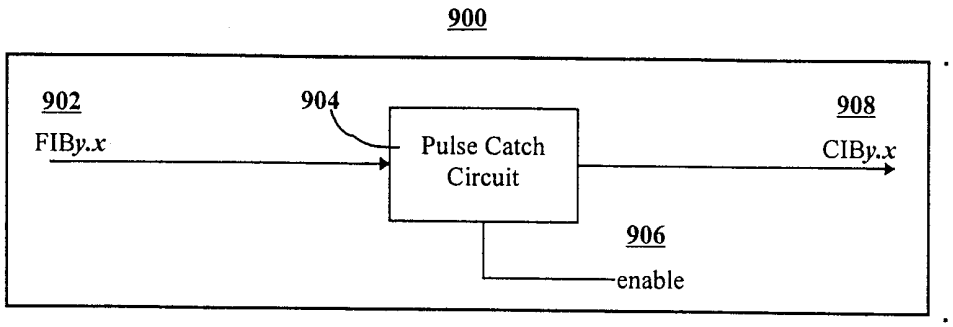


Figure 9

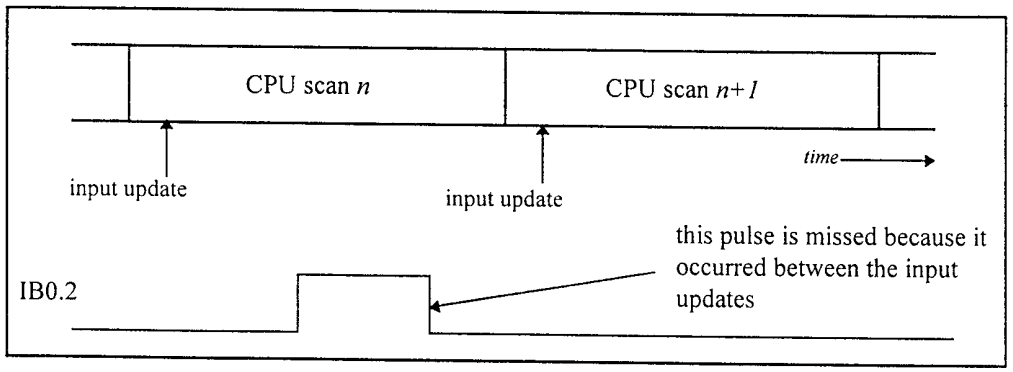


Figure 10

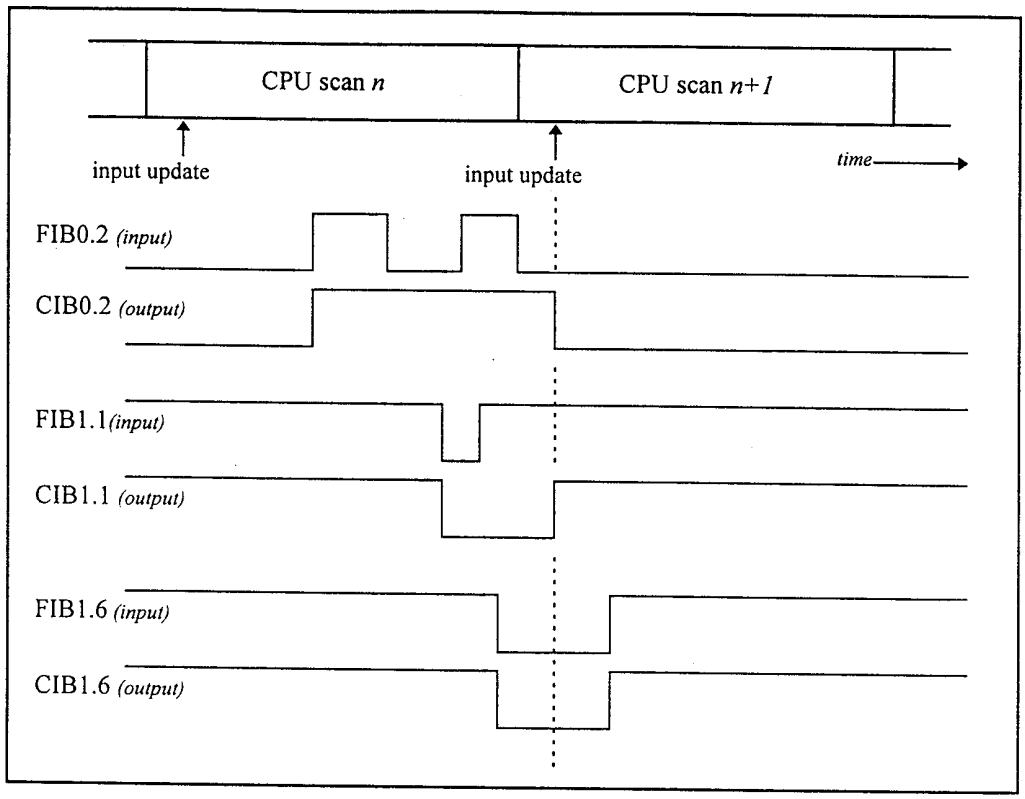

CPU scan n
CPU scan n+1
input update
input update
time
FIB0.2 (input)
CIB0.2 (output)
FIB1.1 (input)
CIB1.1 (output)
FIB1.6 (input)
CIB1.6 (output)

Figure 11

PCE	PS CV	PS F	RP	NS CV	NS F	Comment
1	I	0	-	CV	0	I = CV and F is not set; RP is a don't care
1	not I	0	-	I	1	I ≠ CV; capture new value of I and set F = 1
1	-	1	0	CV	1	CV has not been read
1	-	1	1	I	0	CV has been read; set CV = I
0	-	-	-	I	0	Pulse catch disabled; CV = I

Figure 12a

Pulse Catch Enable Registers

Address	Description
0002H	register name: **IB0_Pulse_Catch_Enable_Register (IB0PCE)** size: **byte (8-bit)** access: **read / write** reset value: **00H** 7 … 0 **EN7 EN6 EN5 EN4 EN3 EN2 EN1 EN0** **ENx:** 1 = enables pulse catch operation on input point IB0.x 0 = disables pulse catch operation on input point IB0.x
0003H	register name: **IB1_Pulse_Catch_Enable_Register (IB1PCE)** size: **byte (8-bit)** access: **read / write** reset value: **00H** 7 … 0 x x **EN5 EN4 EN3 EN2 EN1 EN0** **ENx:** 1 = enables pulse catch operation on input point IB1.x 0 = disables pulse catch operation on input point IB1.x

Figure 12b

register **IB0PS:**	Read of this register returns **CIB0[7:0]** and retriggers pulse catch circuits for IB0 input points	used by SW for input update
register **IB1PS:**	Read of this register returns **CIB1[5:0]** and retriggers pulse catch circuits for IB1 input points	used by SW for input update
register **IB0PSNR:**	Read of this register returns **CIB0[7:0]** and leaves pulse catch circuits unaffected	used by SW for immediate access
register **IB1PSNR:**	Read of this register returns **CIB1[5:0]** and leaves pulse catch circuits unaffected	used by SW for immediate access

Figure 12c

Input Point Status Registers

Address	Description
0004H	register name: **IB0_Input_Point_Status_Register (IB0PS)** size: **byte (8-bit)** access: **read only** reset value: **00H** 7 … 0 CI7 | CI6 | CI5 | CI4 | CI3 | CI2 | CI1 | CI0 **CIx:** Contains conditioned input point state CIB0.x
0005H	register name: **IB1_Input_Point_Status_Register (IB1PS)** size: **byte (8-bit)** access: **read only** reset value: **00H** 7 … 0 x | x | CI5 | CI4 | CI3 | CI2 | CI1 | CI0 **CIx:** Contains conditioned input point state CIB1.x.
0006H	register name: **IB0_Input_Point_Status_Register_No_Retrigger (IB0PSNR)** size: **byte (8-bit)** access: **read only** reset value: **00H** 7 … 0 CI7 | CI6 | CI5 | CI4 | CI3 | CI2 | CI1 | CI0 **CIx:** Contains conditioned input point state CIB0.x
0007H	register name: **IB1_Input_Point_Status_Register_No_Retrigger (IB1PSNR)** size: **byte (8-bit)** access: **read only** reset value: **00H** 7 … 0 x | x | CI5 | CI4 | CI3 | CI2 | CI1 | CI0 **CIx:** Contains conditioned input point state CIB1.x.

Figure 12d

1200e

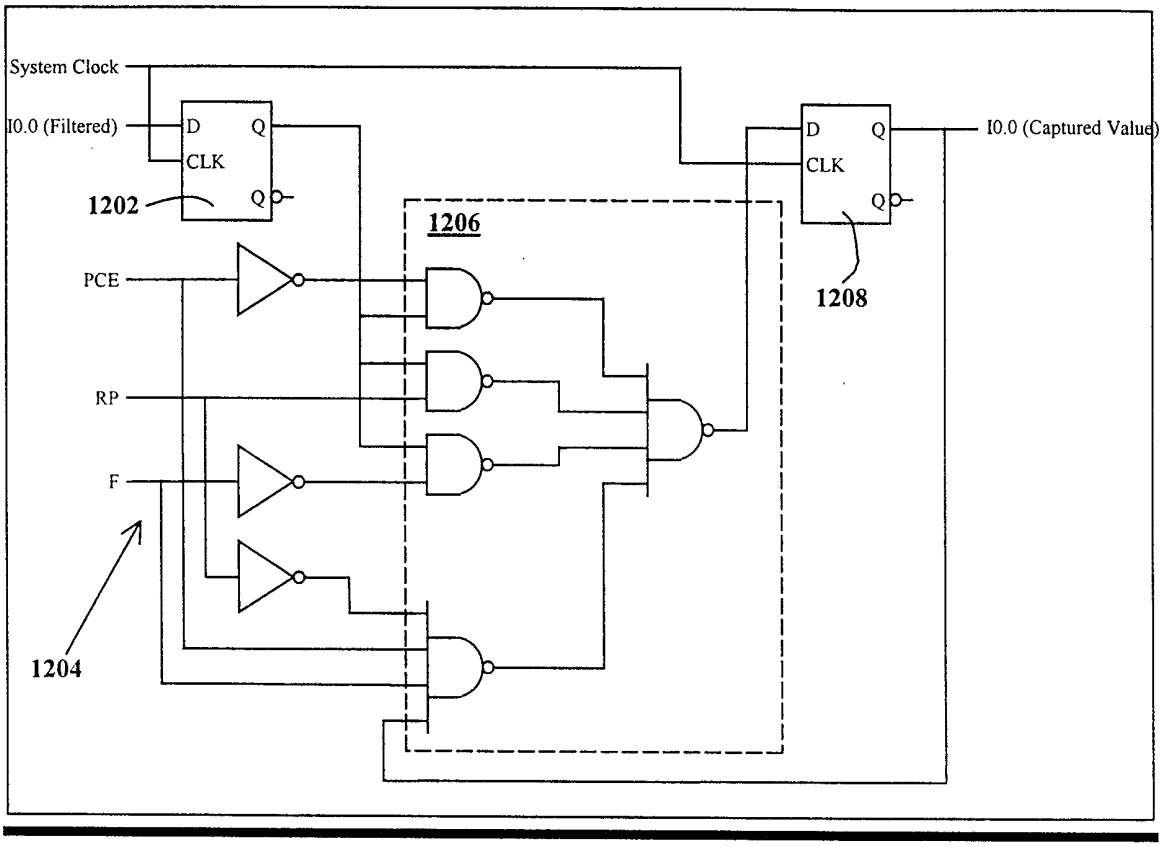


Figure 12e

1200f

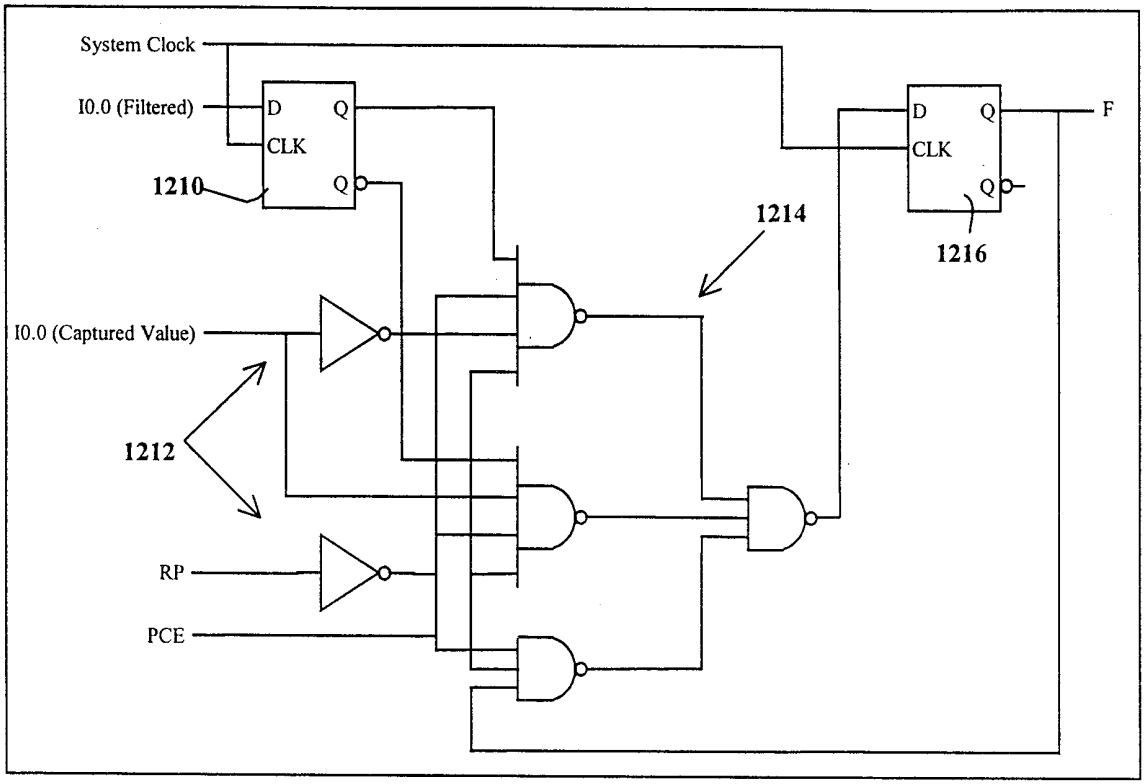


Figure 12f

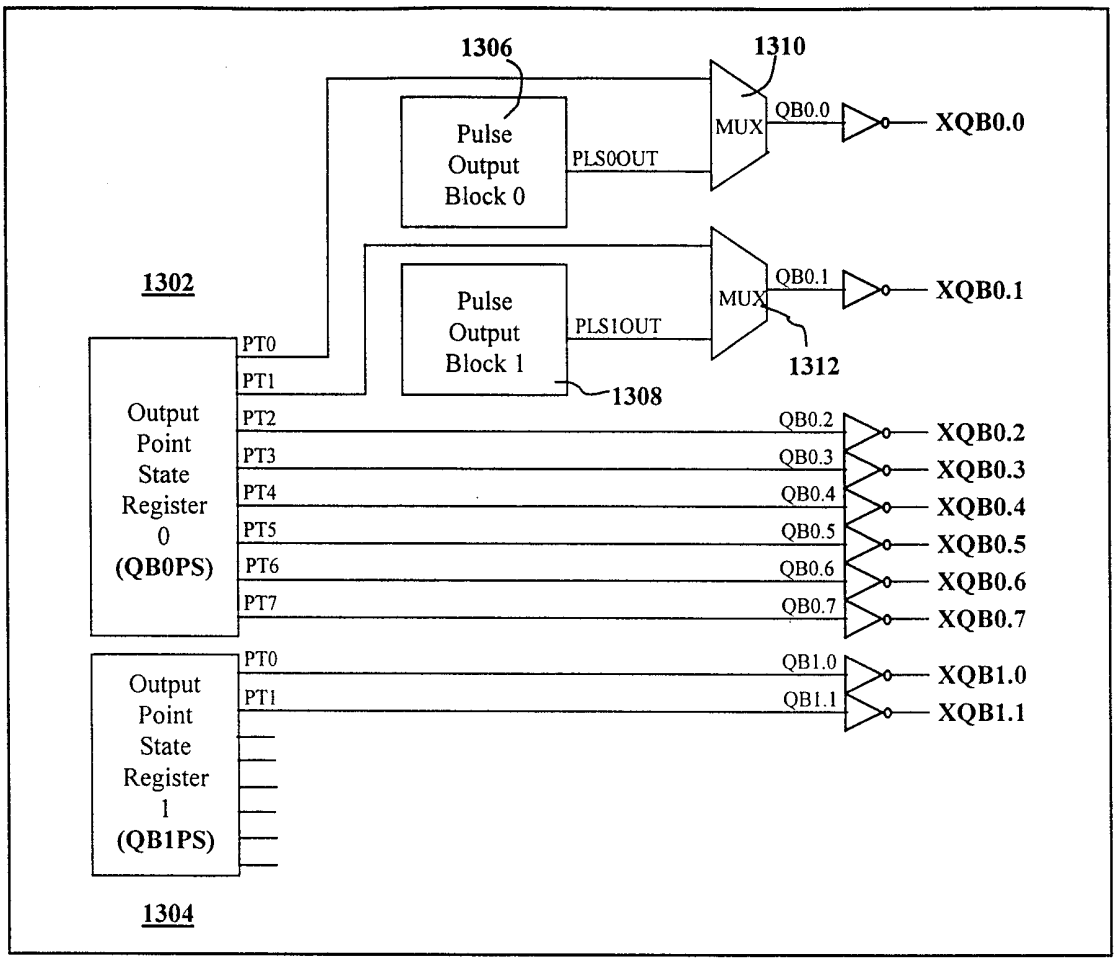

1306
1310
Pulse Output Block 0
PLS0OUT
MUX
QB0.0
XQB0.0
1302
Pulse Output Block 1
PLS1OUT
MUX
QB0.1
XQB0.1
1312
1308
Output Point State Register 0 (QB0PS)
PT0
PT1
PT2
PT3
PT4
PT5
PT6
PT7
QB0.2
QB0.3
QB0.4
QB0.5
QB0.6
QB0.7
XQB0.2
XQB0.3
XQB0.4
XQB0.5
XQB0.6
XQB0.7
Output Point State Register 1 (QB1PS)
PT0
PT1
QB1.0
QB1.1
XQB1.0
XQB1.1
1304

Figure 13

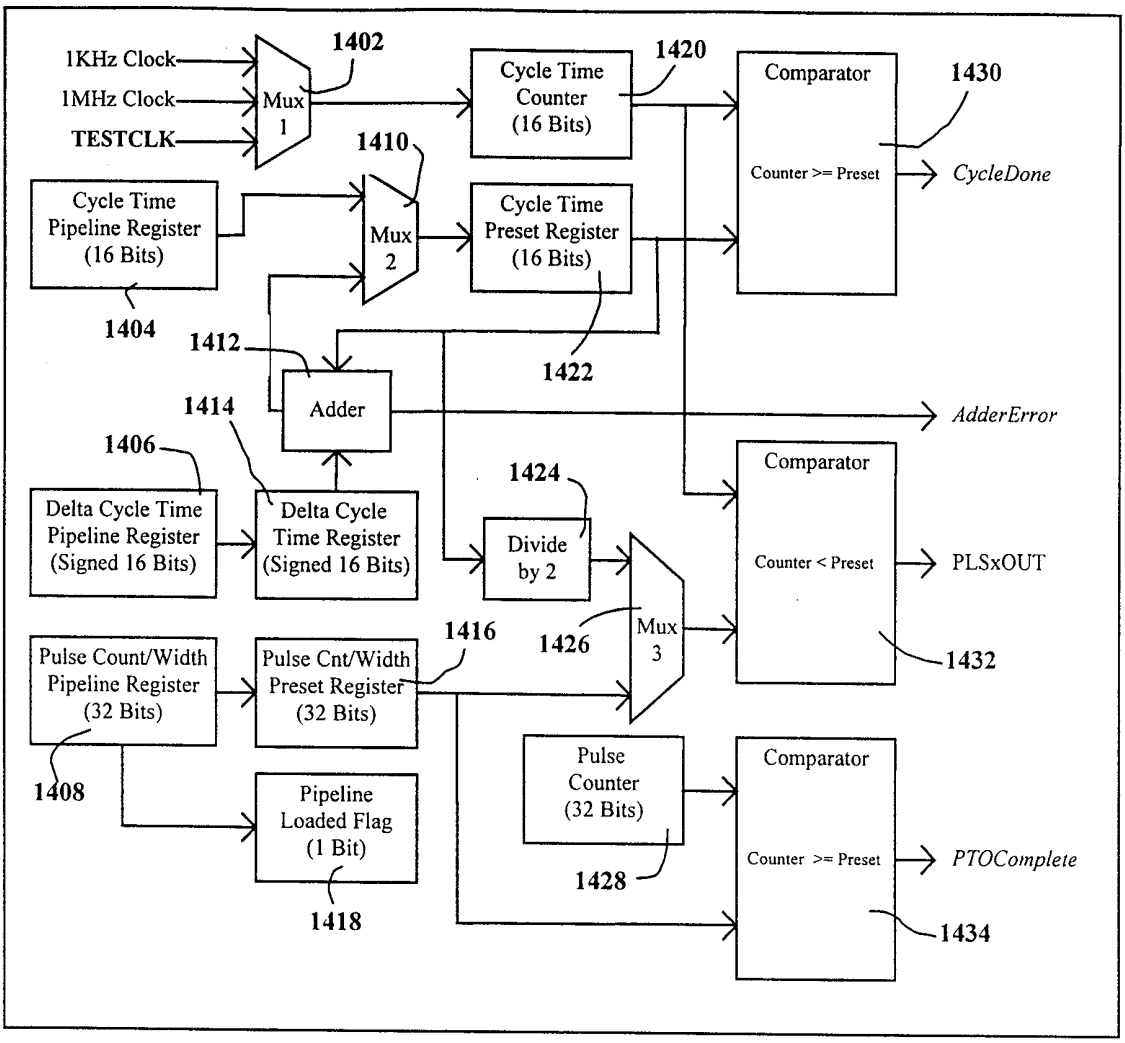

1KHz Clock
1MHz Clock
TESTCLK
Mux 1
1402
Cycle Time Counter (16 Bits)
1420
Comparator
1430
Counter >= Preset
CycleDone
1410
Mux 2
Cycle Time Pipeline Register (16 Bits)
1404
Cycle Time Preset Register (16 Bits)
1422
1412
Adder
AdderError
1414
1406
Delta Cycle Time Pipeline Register (Signed 16 Bits)
Delta Cycle Time Register (Signed 16 Bits)
1424
Divide by 2
Comparator
Counter < Preset
PLSxOUT
1426
Mux 3
1432
1416
Pulse Count/Width Pipeline Register (32 Bits)
1408
Pulse Cnt/Width Preset Register (32 Bits)
Pulse Counter (32 Bits)
1428
Comparator
Counter >= Preset
PTOComplete
Pipeline Loaded Flag (1 Bit)
1418
1434

Figure 14a

Register Name	Valid Value Range
Cycle Time Preset Register Cycle Time Pipeline Register	2 to 65535
Delta Cycle Time Register Delta Cycle Pipeline Register	-32768 to 32767
Pulse Count/Width Preset Register Pulse Count/Width Pipeline Register	1 to (2^{32}-1) 0 to 65535 PTO mode PWM mode

Figure 14b

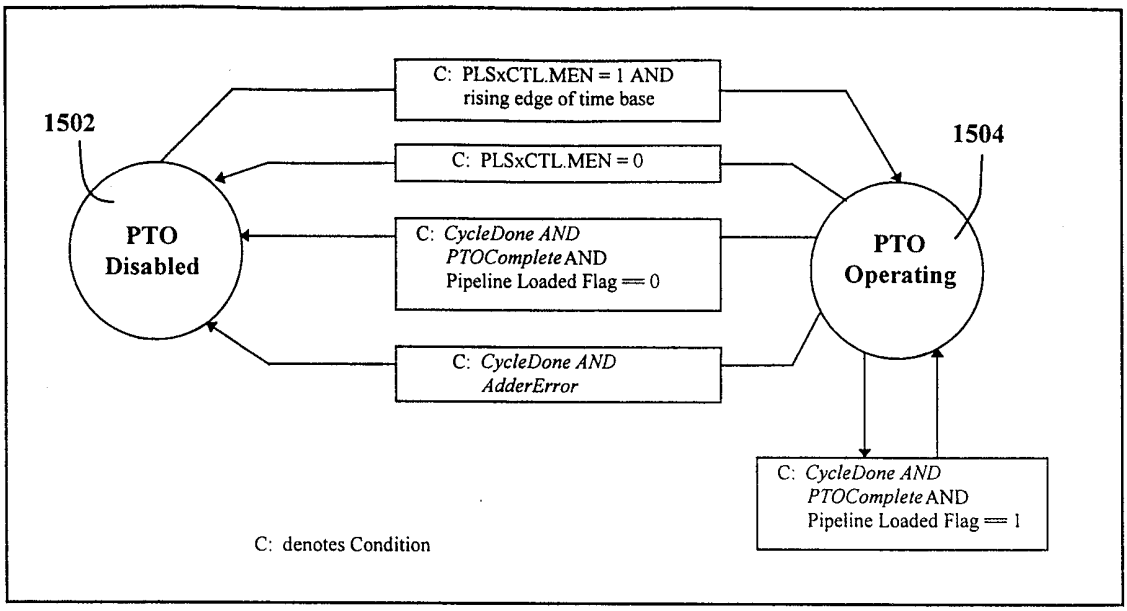

1502
1504
C: PLSxCTL.MEN = 1 AND
rising edge of time base
C: PLSxCTL.MEN = 0
C: CycleDone AND
PTOComplete AND
Pipeline Loaded Flag == 0
C: CycleDone AND
AdderError
PTO
Disabled
PTO
Operating
C: CycleDone AND
PTOComplete AND
Pipeline Loaded Flag == 1
C: denotes Condition

Figure 15

```
cycle time counter = cycle time counter + 1;
IF (cycle time counter >= cycle time preset) THEN
BEGIN  // this is the CycleDone event
        --------
        pulse counter = pulse counter + 1;
        IF (pulse counter >= pulse count preset) THEN
        BEGIN  // this is the PTOComplete event
                --------
                assert INTxPLS signal, if PTOComplete interrupts are enabled;
                IF (pipeline loaded flag is set) THEN
                BEGIN
                        --------
                        transfer values from pipeline registers into operating registers;
                        set pulse counter = 0;
                        clear pipeline loaded flag;
                        --------
                END
                ELSE   // pipeline loaded flag is not set
                BEGIN
                        --------
                        GOTO PTO Disabled state;  // disable the PLS block now
                        --------
                END
                ENDIF
                --------
        END

        ELSE   // not yet at PTOComplete
        BEGIN
                --------
                cycle time preset = cycle time preset + delta cycle time;
                IF (cycle time preset exceeds bounds) THEN
                BEGIN  // this is the AdderError event
                        --------
                        assert INTxPLS signal, if AdderError interrupts are enabled;
                        GOTO PTO Disabled state;  // disable the PLS block now
                        --------
                END
                ENDIF
                --------
        END
        ENDIF
        set cycle time counter = 0;
        --------
END
ENDIF

IF (cycle time counter >= (1/2 * cycle time preset)) THEN
        PLSxOUT = 0;
ELSE   // output still in logic high portion of the current cycle
        PLSxOUT = 1;
ENDIF
```

Figure 16

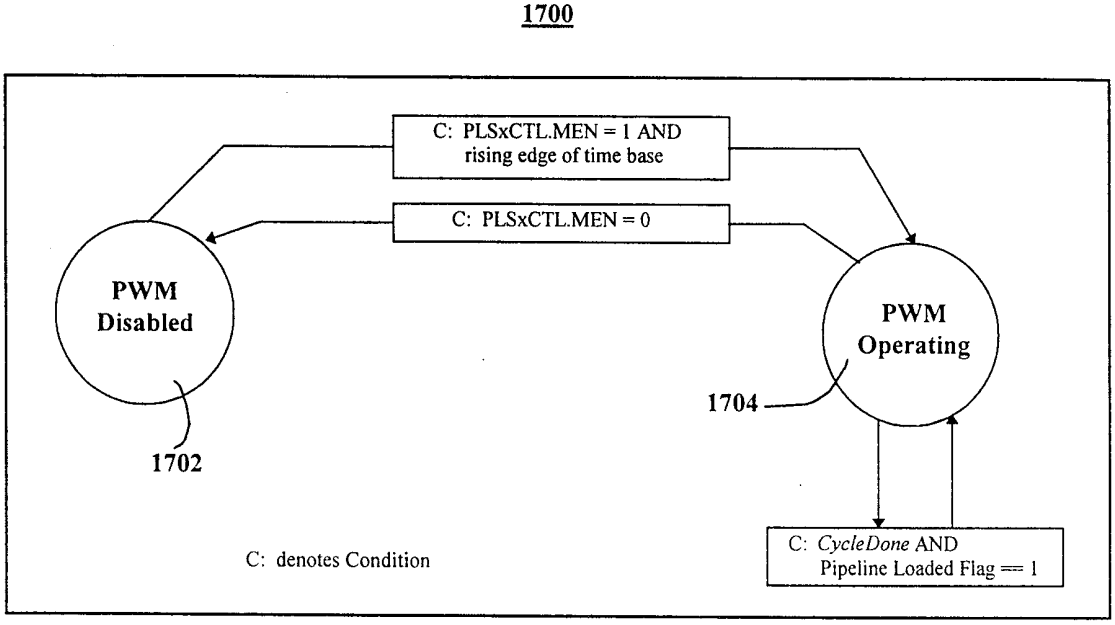

1700
C: PLSxCTL.MEN = 1 AND
rising edge of time base
C: PLSxCTL.MEN = 0
PWM
Disabled
1702
PWM
Operating
1704
C: CycleDone AND
Pipeline Loaded Flag == 1
C: denotes Condition

Figure 17

```
cycle time counter = cycle time counter + 1;
IF (cycle time counter >= cycle time preset) THEN
BEGIN  // this is the CycleDone event
        --------
        IF (pipeline loaded flag is set) THEN
        BEGIN
                --------
                transfer values from pipeline registers into operating registers;
                clear pipeline loaded flag;
                --------
        END
        ENDIF
        set cycle time counter = 0;
        --------
END
ENDIF

IF (cycle time counter >= (pulse width preset)) THEN
        PLSxOUT = 0;
ELSE    // output still in logic high portion of the current cycle
        PLSxOUT = 1;
ENDIF
```

Figure 18

HIGH SPEED OPERATIONS				
Instruction	Mnemonic & Operand(s)	Description	STL Status Element	VALID OPERANDS
Pulse Train Output Profile	PTOP t, n	When S0 = 1, the PTO profile specified in the table for output n is executed. ENO ← S0 * /e	STK, <current step>	Enable: S0 Table: VB, IB, QB, MB, (UI) SMB, SB, LB, *VD, *AC Output: KW (UI) 0 - 1

Definition of the TABLE for PTOP:

Byte Offset	Segment	Description of Table Entries
0		Number of profile segments (40 segments maximum)
1		Current step number being executed
2	#1	Number of steps (4 steps minimum)
4		Starting cycle time for this segment (2 to 65535 μsec)
6		Change in cycle time per step (signed value) (0 to 65535 μsec)
8	#2	Number of steps (4 steps minimum)
10		Starting cycle time for this segment (2 to 65535 μsec)
12		Change in cycle time per step (signed value) (0 to 65535 μsec)
:	:	:
:	:	:

Figure 19

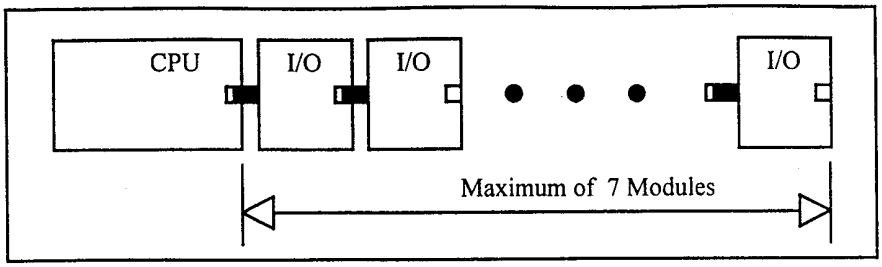


Figure 20

Signal Name	Use
EMD	Expansion Module Data - a bi-directional signal used to communicate the address and data to and from the module.
EMC[1:0]	Expansion Module Clocks - One clock is used to access expansion I/O external to the PLC, while the other is used to access I/O that is local to the PLC that does not connect directly to the ASIC.
XA_OD	Address/Output Disable - a dual function signal used to reset the state machine in the modules on the first clock of each access cycle (active low) and used to indicate output disable when a fatal error has been detected (active low for an RC time constant).
EMDDIR	Expansion Module Data Direction - this signal indicates the direction of data flow on the EMD signal line. 0 - Data is driven by the module to the PLC 1 - Data is driven by the PLC to the module
EMA[2:0]	Expansion Module Address - these signals are daisy chained from PLC to module to module. The value input to a module becomes that module's address. The module will output its address plus one to the next module. The PLC drives these signals to 0 so that the module connected to the PLC has the address of 0.
+5V	5 volt power supply - Two signals carry +5 volts.
GND	Power supply return - Two signals carry ground.

Figure 21

2200

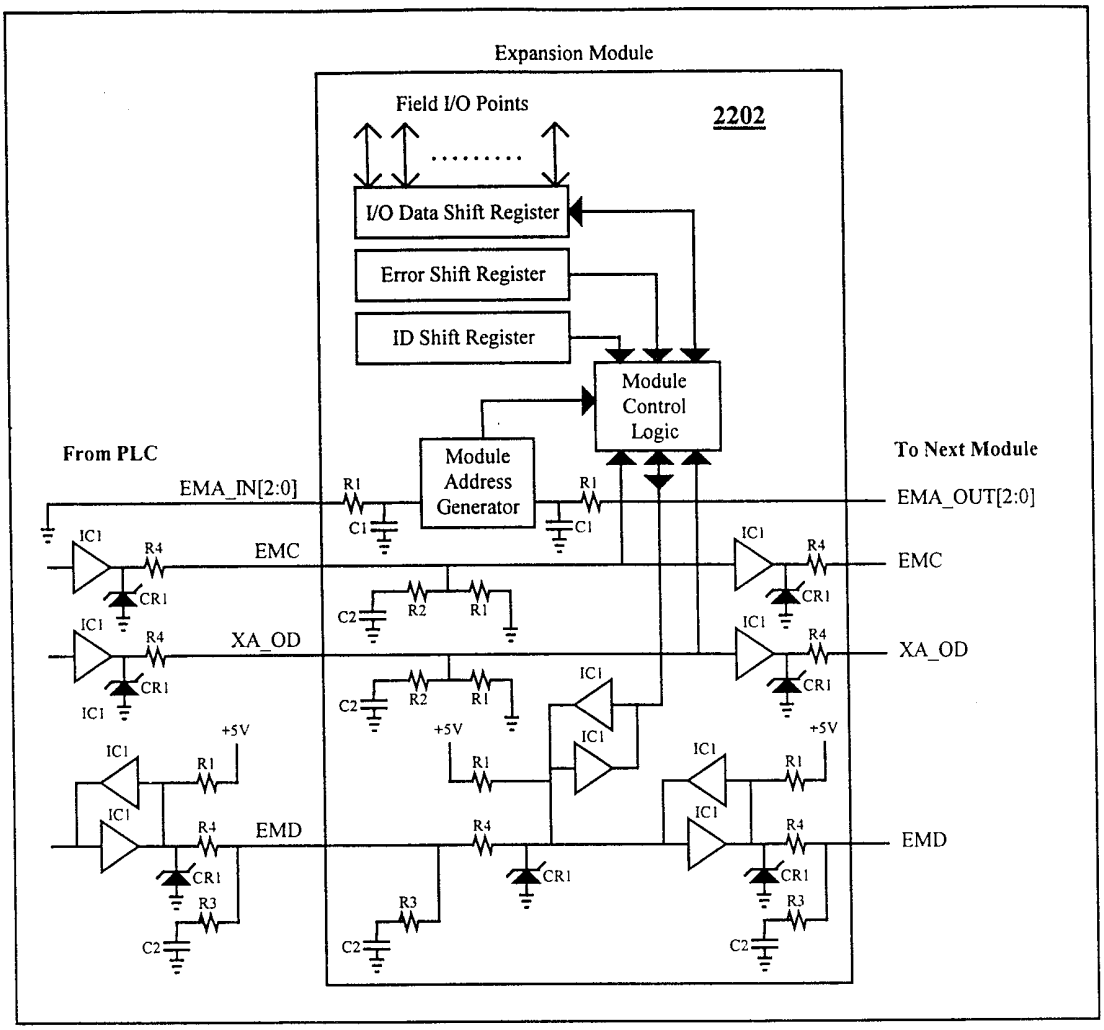


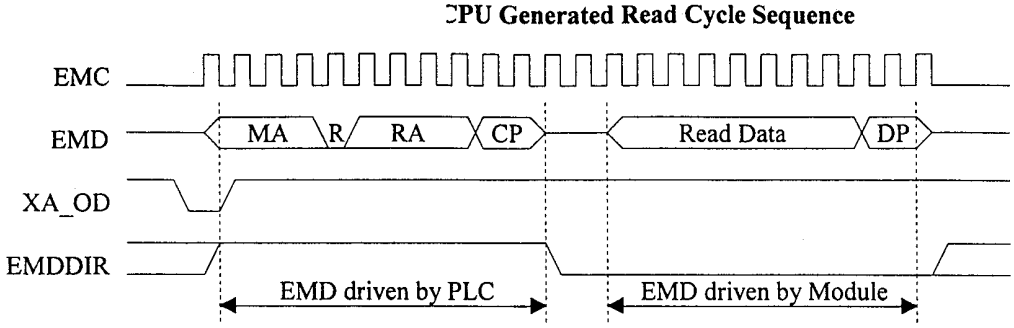
Figure 22

Reference Designator	Component Value/Type	Component Description
IC1	74ABT125/74ABT126	Tri-state buffer
CR1	TVS 5.6V Zener	Diode
R1	4.7K ohm	Resistor
R2	110 ohm	Resistor
R3	220 ohm	Resistor
R4	22 ohm	Resistor
C1	0.1μF	Capacitor
C2	100pF	Capacitor

Figure 23

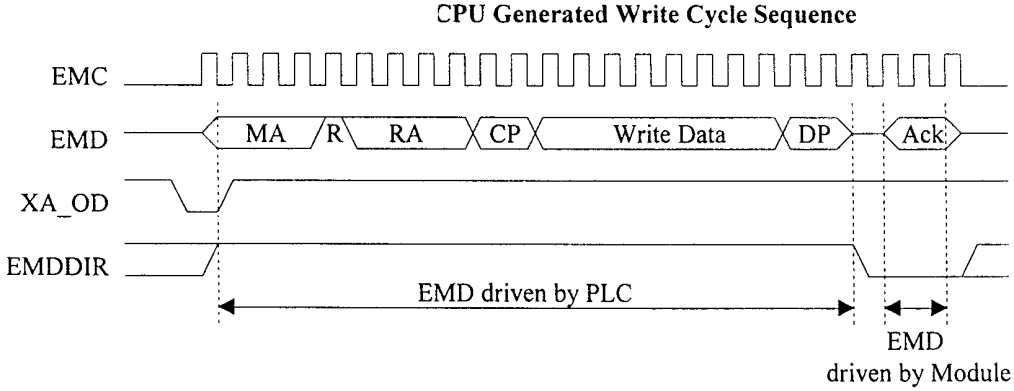
Signal Name	Driving Device	Drive Levels				Receive Levels	
		V_{OL} (Volts) (Max)	I_{OL} (mA) (Min)	V_{OH} (Volts) (Max)	I_{OH} (mA) (Min)	V_{IL} (Volts) (Max)	V_{IH} (Volts) (Min)
EMA[2:0]	Module	0.5	3.0	2.4	-3.0	0.8	2.0
XA_OD	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
EMC	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
EMD	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
	Module	0.55	64.0	2.0	-32.0	0.8	2.0

Figure 24



MA - Module Address for modules 0 to 6 (3 bits)
R - Read/Write bit (Read active low)
RA - Register Address for registers R0 to R15 (4 bits)
CP - Control Parity generated by the CPU on MA, R, and RA (2 bits)
Read Data - Data read from the Module (8 bits)
DP - Data Parity generated by the module on read data (2 bits)

Figure 25



MA - Module Address for modules 0 to 6 (3 bits)
R - Read/Write bit (Read active low)
RA - Register Address for registers R0 to R15 (4 bits)
CP - Control Parity generated by the CPU on MA, R, and RA (2 bits)
Write Data - Data written to the Module (8 bits)
DP - Data Parity generated by the CPU on write data (2 bits)
Ack - Module acknowledge of a good write cycle (2 bits)

Figure 26

	a	*b*	*c*	*d*	*e*
P0	bit 0	bit 2	bit 4	bit 6	bit 7
P1	bit 1	bit 2	bit 3	bit 5	bit 7
5 bit odd parity = ! (((*a* xor *b*) xor (*c* xor *d*)) xor *e*)					

Figure 27

PLC Type	Port 0	Port 1
CPU 212	256	-
CPU 214	256	-
CPU 215/216	256	256

Figure 28

INTERRUPTS				
Instruction	Mnemonic & Operand(s)	Description	STL Status Elements	VALID OPERANDS
Pass Token	PASS	User program has completed its use of the token hold period and is returning control to the system.	STK	

Figure 29

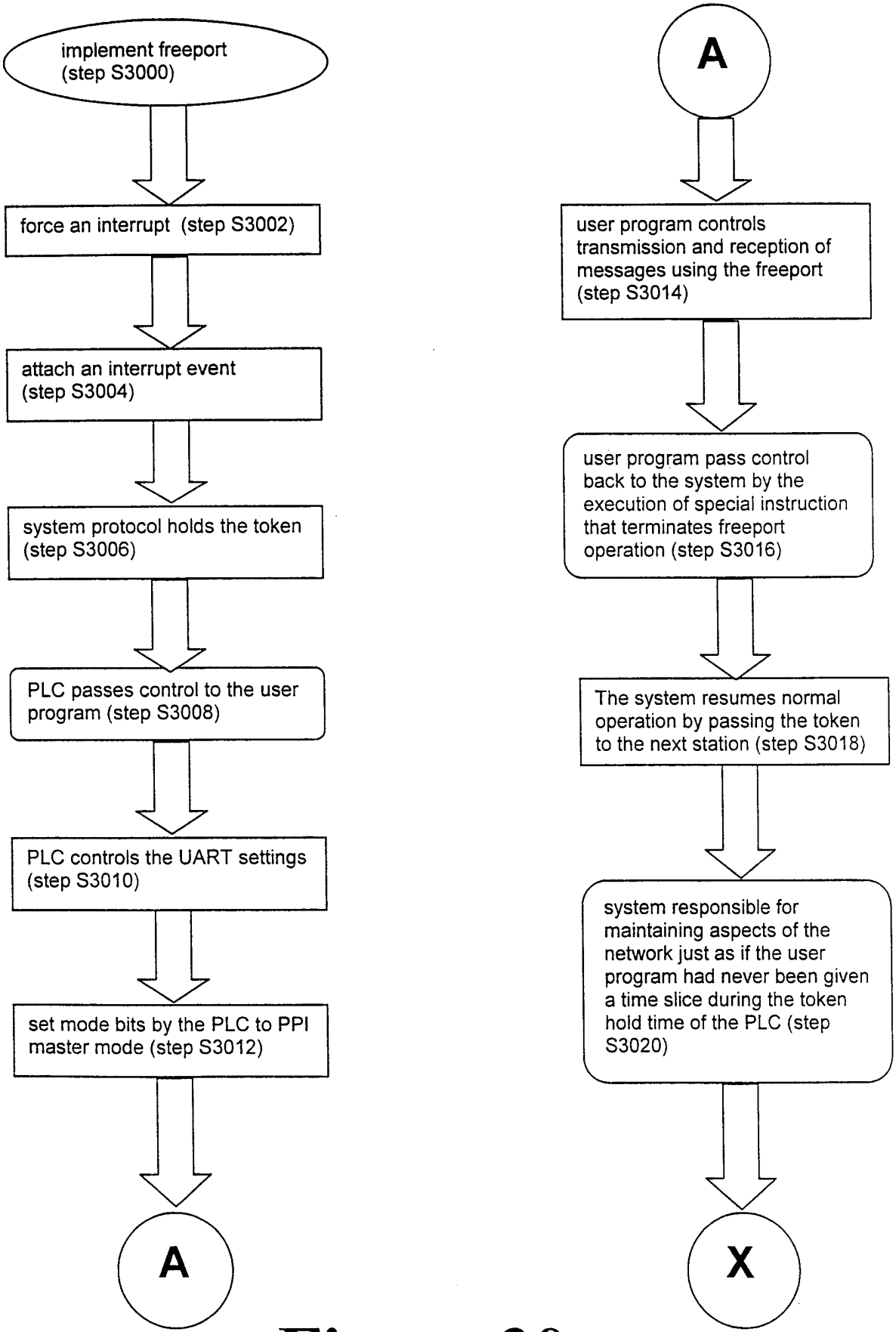

implement freeport (step S3000)
force an interrupt (step S3002)
attach an interrupt event (step S3004)
system protocol holds the token (step S3006)
PLC passes control to the user program (step S3008)
PLC controls the UART settings (step S3010)
set mode bits by the PLC to PPI master mode (step S3012)
A
A
user program controls transmission and reception of messages using the freeport (step S3014)
user program pass control back to the system by the execution of special instruction that terminates freeport operation (step S3016)
The system resumes normal operation by passing the token to the next station (step S3018)
system responsible for maintaining aspects of the network just as if the user program had never been given a time slice during the token hold time of the PLC (step S3020)
X

Figure 30a

<table>
<tr><th colspan="2">SM Bit Definition (Read/Write)</th></tr>
<tr><th>SM Bits</th><th>Description</th></tr>
<tr><td>SM30</td><td>Port 0: Communication port usage

MSB 7 LSB 0
SMB30 | p | p | d | r | r | r | m | m |

pp (Parity): '00' - no parity; '01' - even parity; '10' - no parity; '11' - odd parity
d (Data bits/char): '0' - 8 bits/char; '1' - 7 bits/char
rrr (Baud Rate): '000' - 38,400; '001' - 19,200; '010' - 9600; '011' - 4800; '100' - 2400; '101' - 1200; '110' - 600; '111' - 300
mm (Protocol): '00' - PPI Slave (default); '01' - Freeport; '10' - PPI Master; '11' - reserved (PPI Slave)

When the user selects the code mm = 10 (PPI Master), the PLC will become a master on the network allowing the NETR and NETW instructions to be executed. Bits 2 through 7 are ignored in PPI modes. In PPI Master mode with the token acquired interrupt enabled, these bits are used to setup the UART prior to transferring control to the user's program.</td></tr>
<tr><td>SM130</td><td>Port 1: Communication port usage (CPU 216 only)

MSB 7 LSB 0
SMB130 | p | p | d | r | r | r | m | m |

pp (Parity): '00' - no parity; '01' - even parity; '10' - no parity; '11' - odd parity
d (Data bits/char): '0' - 8 bits/char; '1' - 7 bits/char
rrr (Baud Rate): '000' - 38,400; '001' - 19,200; '010' - 9600; '011' - 4800; '100' - 2400; '101' - 1200; '110' - 600; '111' - 300
mm (Protocol): '00' - PPI Slave (default); '01' - Freeport; '10' - PPI Master; '11' - reserved (PPI Slave)

When the user selects the code mm = 10 (PPI Master), the PLC will become a master on the network allowing the NETR and NETW instructions to be executed. Bits 2 through 7 are ignored in PPI modes. In PPI Master mode with the token acquired interrupt enabled, these bits are used to setup the UART prior to transferring control to the user's program.</td></tr>
</table>

Figure 30b

<table>
<tr><th colspan="2">SM Bit Definition (Read/Write) (continued)</th></tr>
<tr><th>SM Bits</th><th>Description</th></tr>
<tr><td>SM87</td><td>Port 0: Receive message control byte. (CPU 212/214/216)

MSB 7 LSB 0
<table>
<tr><td>en</td><td>sc</td><td>ec</td><td>il</td><td>c/m</td><td>tmr</td><td>bk</td><td>0</td></tr>
</table>

en: Enable/disable receive message bit is checked each time the RCV instruction is executed. If this bit is a "0", then the receive message function is disabled. If this bit is a "1", then the receive message function is enabled.

sc: 0 - ignore SMB88; 1 - use the value of SMB88 to detect start of message

ec: 0 - ignore SMB89; 1 - use the value of SMB89 to detect end of message

il: 0 - ignore SMW90; 1 - use the value of SMW90 to detect an idle line condition[1]

c/m: 0 - use timer as an inter-character timer; 1 - use timer as a message timer

tmr: 0 - ignore SMW92; 1 - terminate receive if the time period in SMW92 is exceeded

bk: 0 - ignore break conditions; 1 - use break condition as start of message detection

[1]By setting the sc and bk bits to 0 and the en, il, c/m and tmr bits to 1 with an idle line timer value of zero, SMW92 will be used to time out the RCV instruction without receiving any characters. If the timer is not used (tmr = 0), then any character received will be used as start of message.

The bits of the message interrupt control byte are used to define the criteria by which the message is identified. Both start of message and end of message criteria are defined. To determine the start of a message either of two sets of logically Anded start of message criteria must be true and must occur in sequence (idle line followed by start character or break followed by start character). To determine the end of a message the enabled end of message criteria are logically ORed. The equations for start and stop criteria are given below:

Start of Message = il * sc + bk * sc

End of Message = ec + tmr + maximum character count reached

Note: Receive will automatically be terminated by an overrun or a parity error (if enabled).</td></tr>
<tr><td>SM88</td><td>Port 0: Start of message character. (CPU 212/214/216)</td></tr>
<tr><td>SM89</td><td>Port 0: End of message character. (CPU 212/214/216)</td></tr>
<tr><td>SM90
SM91</td><td>Port 0: Idle line time period given in milliseconds. The first character received after the idle line time has expired is the start of a new message. SM90 is MSB. (CPU 212/214/216)</td></tr>
<tr><td>SM92
SM93</td><td>Port 0: Inter-character/message timer timeout value given in milliseconds. If the time period is exceeded, the receive message is terminated. SM92 is MSB. (CPU 212/214/216)</td></tr>
<tr><td>SM94</td><td>Port 0: Maximum number of characters to be received (1 to 255 bytes) (CPU 212/214/216)</td></tr>
</table>

Figure 30c

<table>
<tr><th colspan="2">SM Bit Definition (Read/Write) (continued)</th></tr>
<tr><th>SM Bits</th><th>Description</th></tr>
<tr><td>SM187</td><td>Port 1: Message interrupt control byte (CPU 216 only)

MSB 7 LSB 0
<table>
<tr><td>en</td><td>sc</td><td>ec</td><td>il</td><td>c/m</td><td>tmr</td><td>bk</td><td>0</td></tr>
</table>

en: Enable/disable receive message bit is checked each time the RCV instruction is executed. If this bit is a "0", then the receive message function is disabled. If this bit is a "1", then the receive message function is enabled.

sc: 0 - ignore SMB188; 1 - use the value of SMB188 to detect start of message

ec: 0 - ignore SMB89; 1 - use the value of SMB189 to detect end of message

il: 0 - ignore SMW190; 1 - use the value of SMW190 to detect an idle line condition[1]

c/m: 0 - use timer as an inter-character timer; 1 - use timer as a message timer

tmr: 0 - ignore SMW192; 1 - terminate receive if the time period in SMW192 is exceeded

bk: 0 - ignore break conditions; 1 - use break condition as start of message detection

[1]By setting the sc and bk bits to 0 and the en, il, c/m and tmr bits to 1 with an idle line timer value of zero, SMW192 will be used to time out the RCV instruction without receiving any characters. If the timer is not used (tmr = 0), then any character received will be used as start of message.

The bits of the message interrupt control byte are used to define the criteria by which the message is identified. Both start of message and end of message criteria are defined. To determine the start of a message either of two sets of logically Anded start of message criteria must be true and must occur in sequence (idle line followed by start character or break followed by start character). To determine the end of a message the enabled end of message criteria are logically Ored. The equations for start and stop criteria are given below:

Start of Message = il * sc + bk * sc

End of Message = ec + tmr + maximum character count reached

Note: Receive will automatically be terminated by an overrun or a parity error (if enabled).</td></tr>
<tr><td>SM188</td><td>Port 1: Start of message character (CPU 216 only)</td></tr>
<tr><td>SM189</td><td>Port 1: End of message character (CPU 216 only)</td></tr>
<tr><td>SM190
SM191</td><td>Port 1: Idle line time period given in milliseconds. The first character received after the idle line time has expired is the start of a new message. SM190 is MSB. (CPU 216 only)</td></tr>
<tr><td>SM192
SM193</td><td>Port 1: Inter-character/message timer timeout value given in milliseconds. If the time period is exceeded, the receive message is terminated. SM192 is MSB. (CPU 216 only)</td></tr>
<tr><td>SM194</td><td>Port 1: Maximum number of characters to be received (1 to 255 bytes) (CPU 216 only)</td></tr>
</table>

Figure 30d

PLC Type	Port 0			Port 1		
	Number of Connections		Buffer Size	Number of Connections		Buffer Size
	Total	Reserved	(Bytes)	Total	Reserved	(Bytes)
CPU 221	4	1 - PG 1 - OP	128/256	-	-	-
CPU 222	4	1 - PG 1 - OP	128/256	-	-	-
CPU 224	4	1 - PG 1 - OP	128/256	-	-	-
CPU 226	4	1 - PG 1 - OP	128/256	4	1 - PG 1 - OP	128/256

Figure 30e

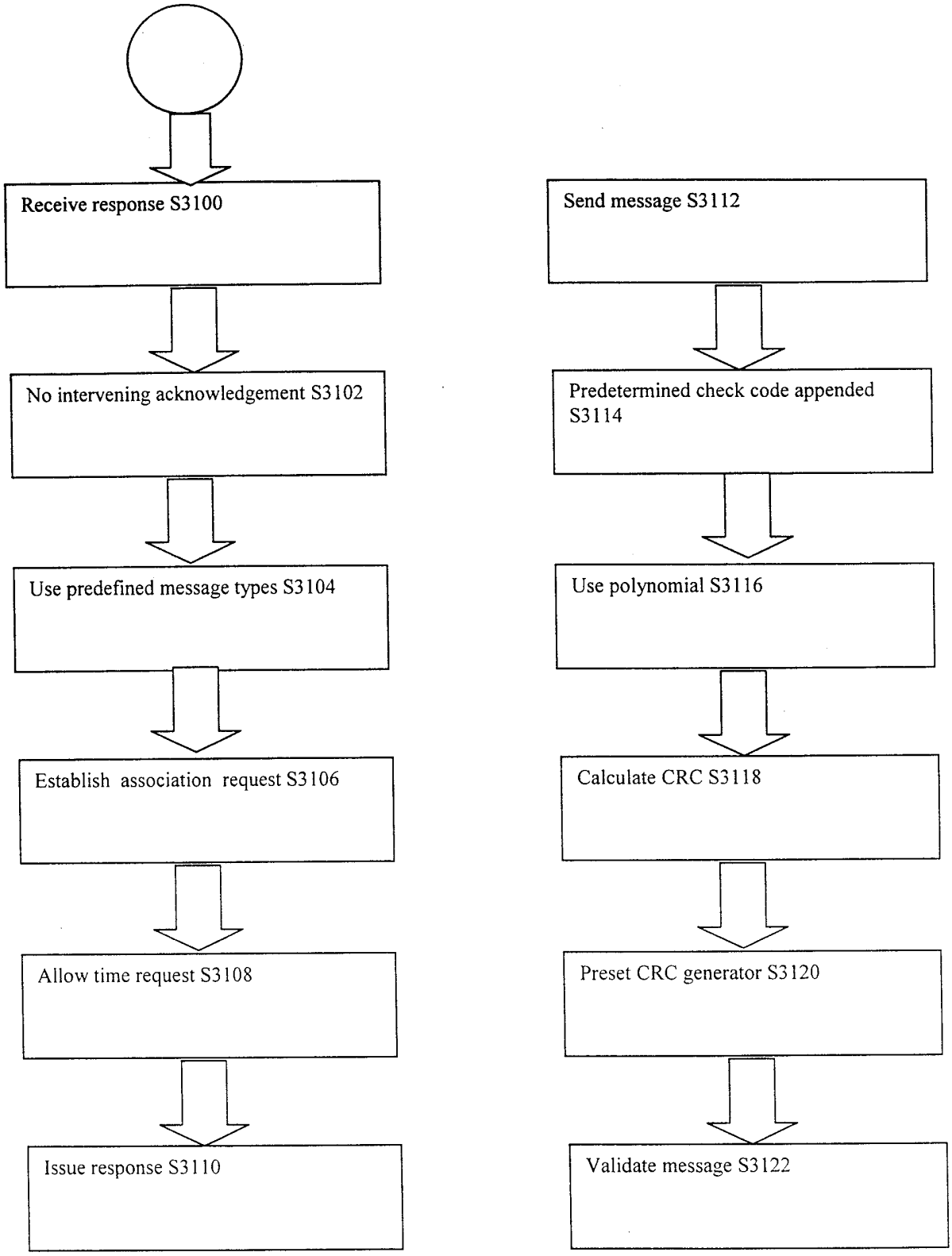

Receive response S3100
No intervening acknowledgement S3102
Use predefined message types S3104
Establish association request S3106
Allow time request S3108
Issue response S3110
Send message S3112
Predetermined check code appended S3114
Use polynomial S3116
Calculate CRC S3118
Preset CRC generator S3120
Validate message S3122

Figure 31

PROGRAM CONTROL FUNCTIONS				
Instruction	Mnemonic & Operand(s)	Description	STL Status Element	VALID OPERANDS
Hide	HIDE n, a, p	The HIDE label marks the start of a block of n instructions that are encrypted using the password, p, when a is non-zero.	STK, SMB1	Enable: None n: KW (2 bytes) (UI) a: KW (2 bytes) (UI) p: KD (4 bytes) (UI)

Figure 32a

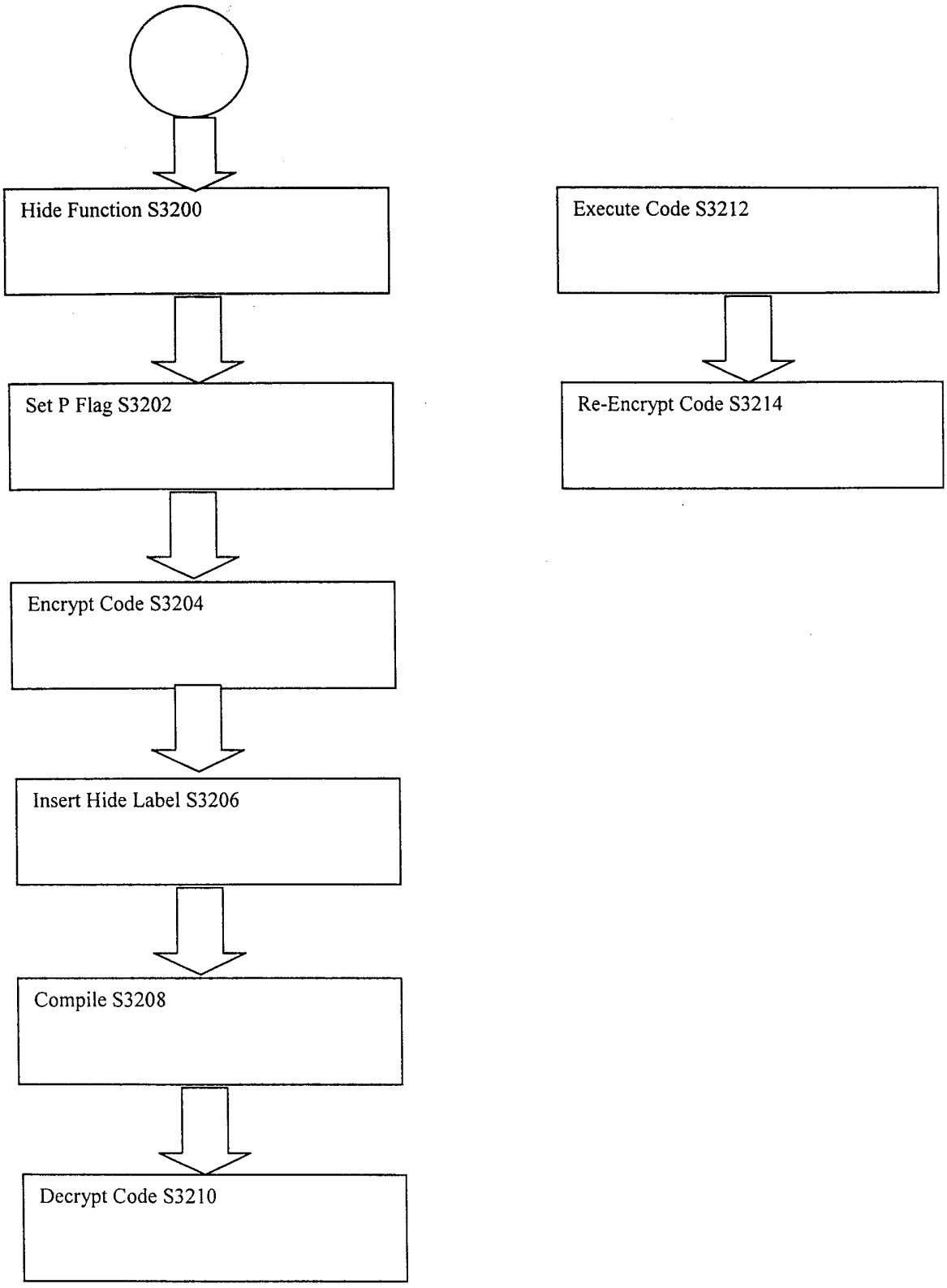

Hide Function S3200
Set P Flag S3202
Encrypt Code S3204
Insert Hide Label S3206
Compile S3208
Decrypt Code S3210
Execute Code S3212
Re-Encrypt Code S3214

Figure 32b

PROGRAM CONTROL FUNCTIONS				
Instruction	Mnemonic & Operand(s)	Description	STL Status Element	VALID OPERANDS
System Function Call	SFC f, s, #_parms, p0, p1, p2, ...	When S0 = 1, execute the system function identified by f and the sub-function identified by s.	STK, SMB1, <p0>, <p1>, <p2>, ...	Enable: S0 f: KW (UI) 0-65536 s: KW (UI) 0-65536 #parms: KB (UI) 0-16 p0, ... : Bit, Byte, Word, Dword Bit V, I, Q, M, SM, S, T, C, L Byte VB, IB, QB, MB, SMB, SB, KB, LB Word VW, T, C, IW, QW, MW, SMW, SW, LW, KW Dword VD, ID, QD, MD, SMD, SD, HC, LD, KD, &VB, &IB, &QB, &MB, &T, &C, &SB Note: Constants and address pointer specifications are not allowed for output or input/output parameters.

Figure 33a

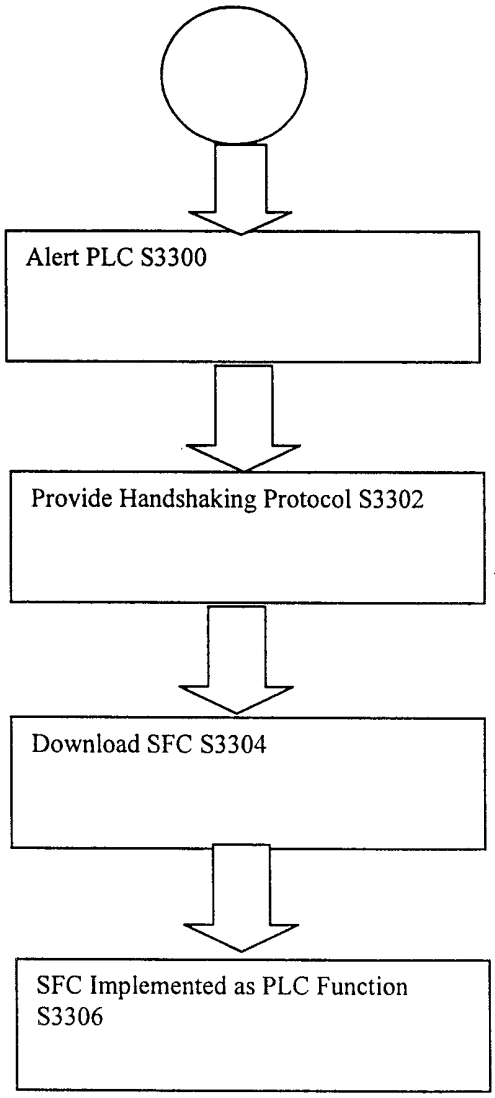

Alert PLC S3300
Provide Handshaking Protocol S3302
Download SFC S3304
SFC Implemented as PLC Function S3306

Figure 33b

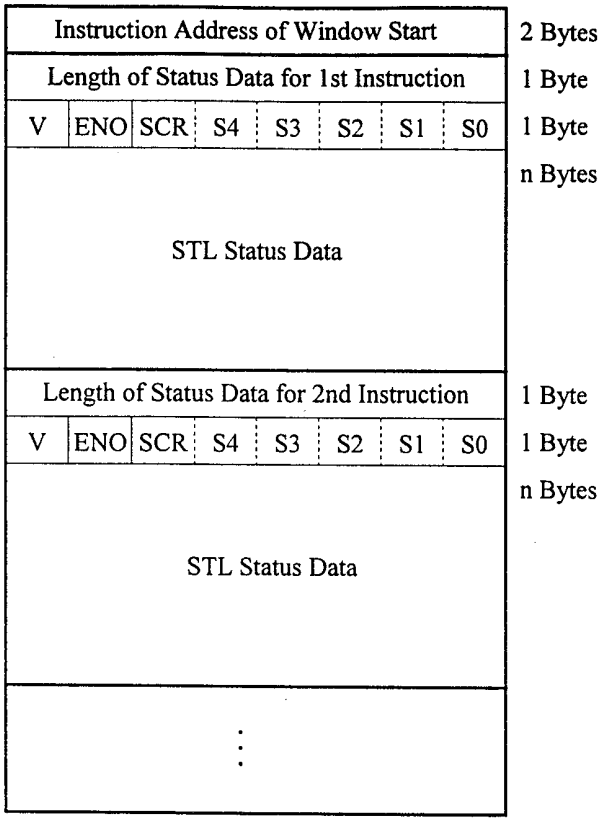


Figure 34a

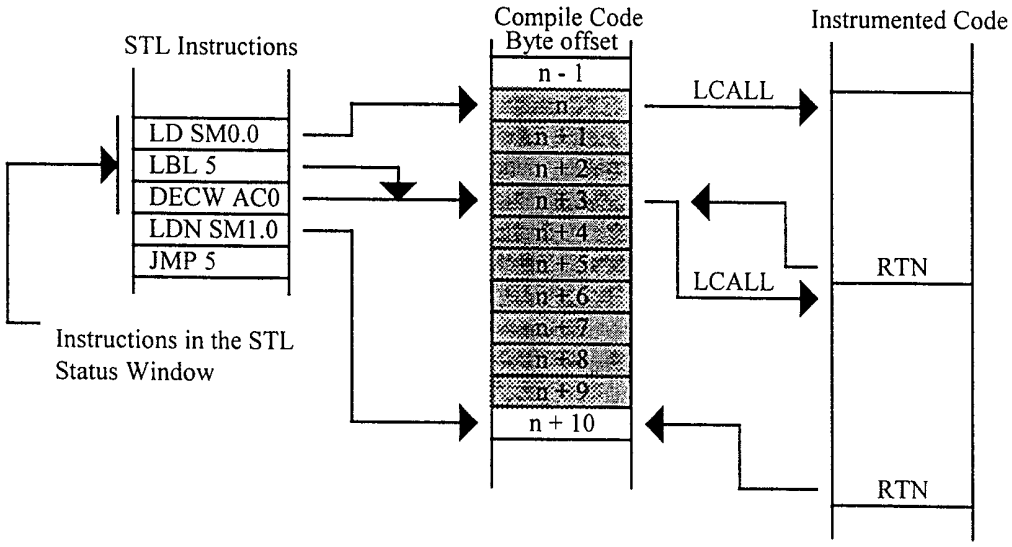


Figure 34b

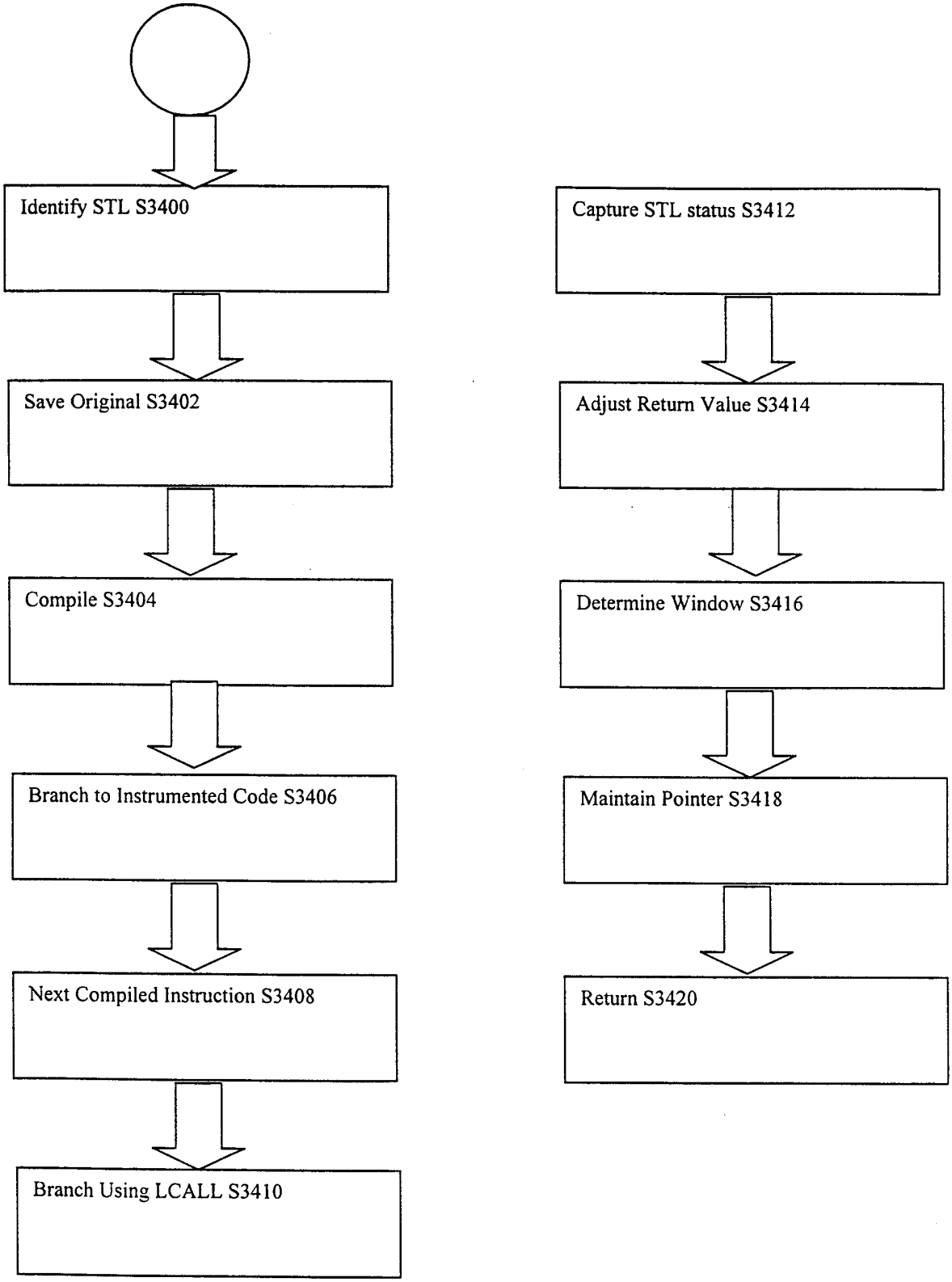

Identify STL S3400
Save Original S3402
Compile S3404
Branch to Instrumented Code S3406
Next Compiled Instruction S3408
Branch Using LCALL S3410
Capture STL status S3412
Adjust Return Value S3414
Determine Window S3416
Maintain Pointer S3418
Return S3420

Figure 34c

	Basic Instr.	Compiled To	Bytes	Cycles
1	LD	RLC A MOV C, <bit>	1 2	1 1
2	LDN	RLC A MOV C, <bit> CPL C	1 2 1	1 1 1
3	A	ANL C, <bit> NOP	2 1	2 1
4	AN	ANL C, /<bit> NOP	2 1	2 1
5	O	ORL C, <bit> NOP	2 1	2 1
6	ON	ORL C, /<bit> NOP	2 1	2 1
7		MOV <bit>, C NOP	2 1	2 1
8	ALD	ANL C, ACC.0 RR A	2 1	2 1
9	OLD	ORL C, ACC.0 RR A	2 1	2 1
10	NOT	CPL C NOP NOP	1 1 1	1 1 1
11	LPS	RL A MOV ACC.0, A	1 2	1 2
12	LPP	RRC A NOP NOP	1 1 1	1 1 1
13	LRD	MOV C, ACC.0	2	2
14	RET	RET NOP NOP	1 1 1	2 1 1
15	INT	CLR A NOP NOP	1 1 1	1 1 1
16	END	JNC $+0 RET	2 1	2 2
17	CRETI	JNC $+0 RET	2 1	2 2

Figure 34d

Housing	Height	Width	Depth
CPU 221	80 mm	90 mm	62 mm
CPU 222	80 mm	90 mm	62 mm
CPU 224	80 mm	120.5 mm	62 mm
CPU 226	80 mm	190 mm	62 mm
8 Point I/O Module	80 mm	46 mm	62 mm
16 Point I/O Module	80 mm	71.2 mm	62 mm
32 Point I/O Module	80 mm	125 mm	62 mm
Intelligent Module	80 mm	90 mm	62 mm

Figure 35a

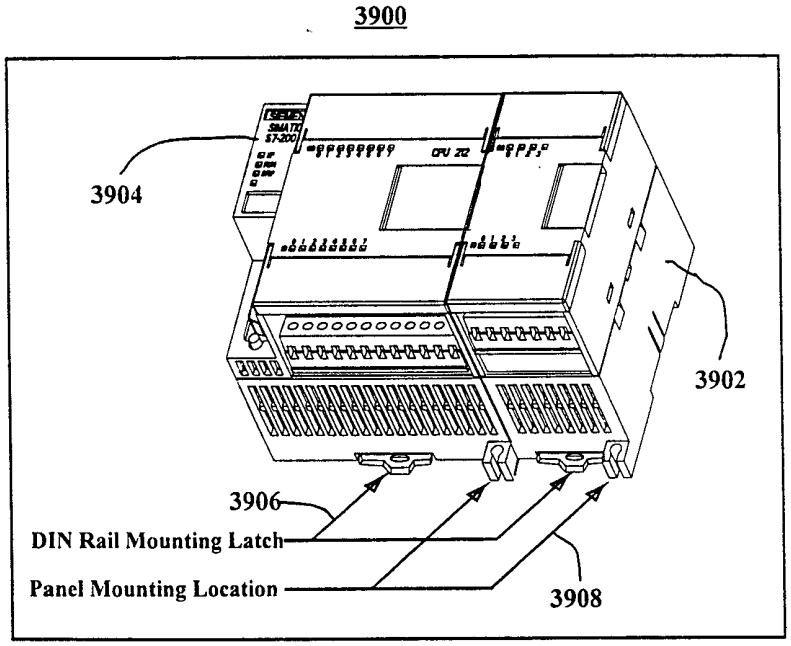


Figure 35b

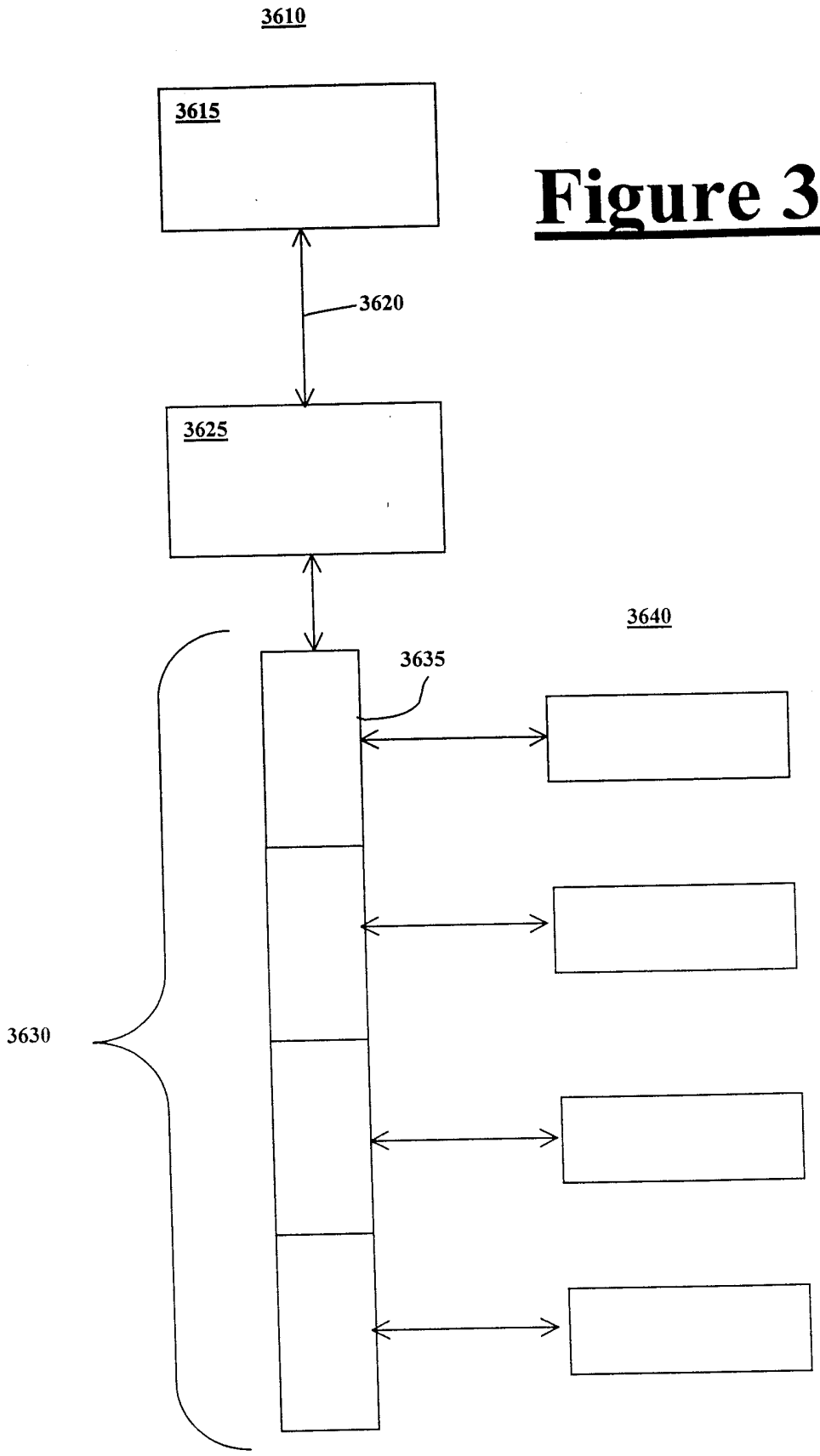


Figure 36

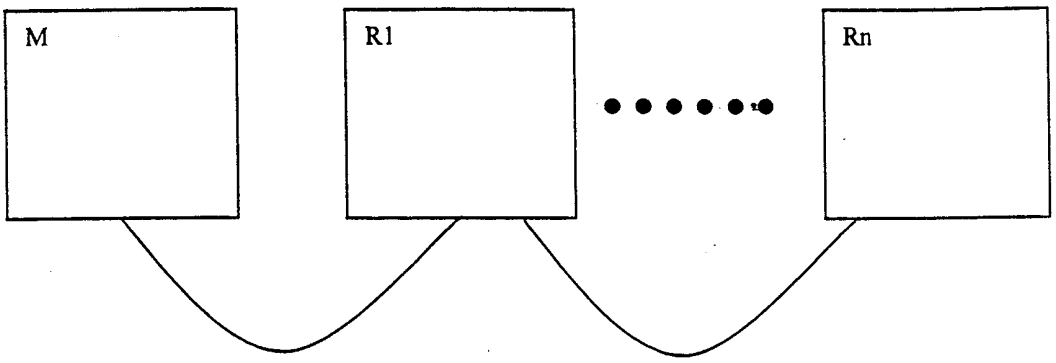


Figure 37a

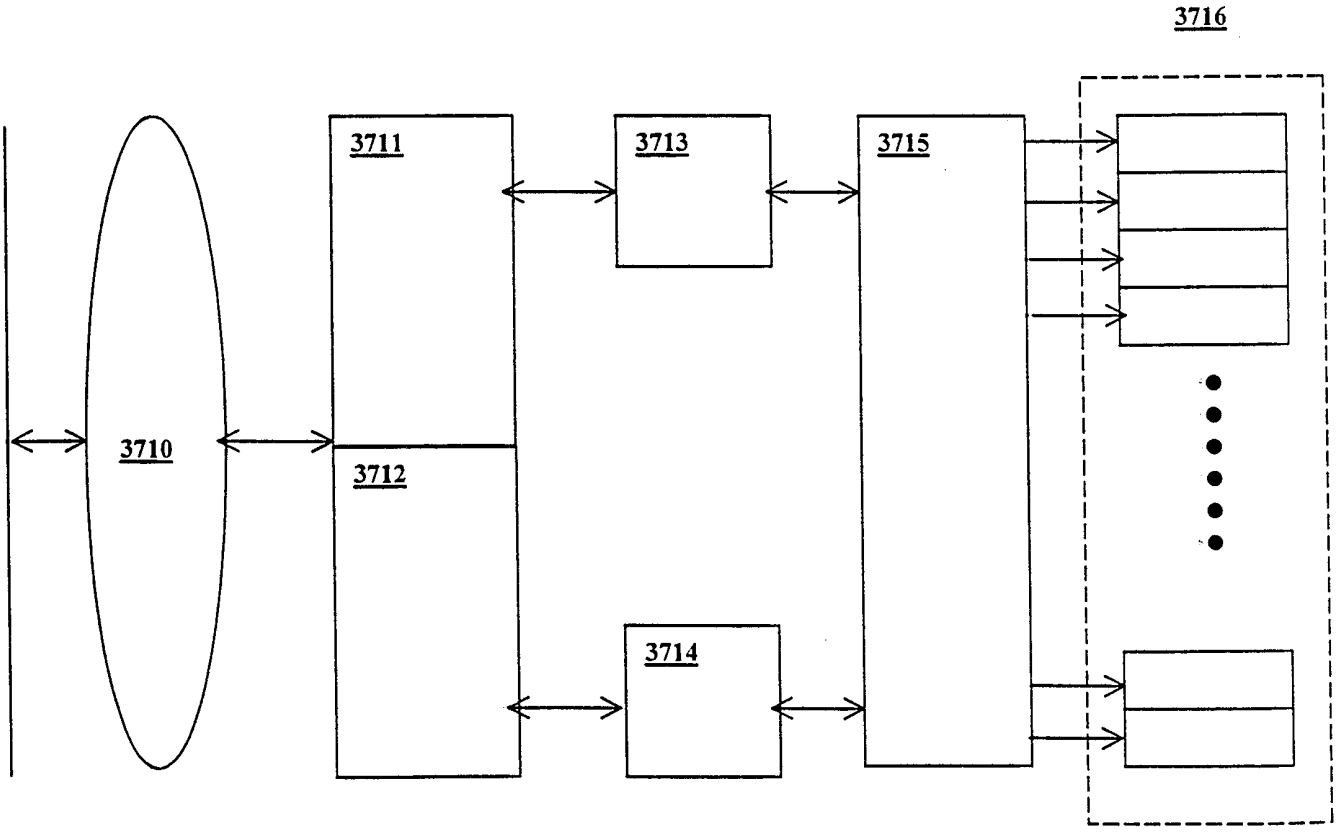


Figure 37b

INTERNATIONAL SEARCH REPORT

Interi nal Application No
PCT/US 00/06796

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G05B19/05

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 7 G05B H03K

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)

EPO-Internal, WPI Data, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 343 317 A (HITACHI LTD) 29 November 1989 (1989-11-29) column 1, line 3 -column 3, line 43 column 4, line 32 -column 7, line 26; figures 1-9	1-20
A	EP 0 105 667 A (XEROX CORP) 18 April 1984 (1984-04-18) page 11, line 1 -page 13, last line; figures 5-7	1-20

☐ Further documents are listed in the continuation of box C.

☒ 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
6 July 2000	14/07/2000

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	Nettesheim, J

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

INTERNATIONAL SEARCH REPORT

information on patent family members

Interr ıal Application No
PCT/US 00/06796

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0343317 A	29-11-1989	JP 1297913 A	01-12-1989
		JP 2585372 B	26-02-1997
		DE 68910379 D	09-12-1993
		DE 68910379 T	14-04-1994
		ES 2045209 T	16-01-1994
		US 4961014 A	02-10-1990
EP 0105667 A	18-04-1984	US 4550382 A	29-10-1985
		CA 1213308 A	28-10-1986
		DE 3375954 D	14-04-1988
		ES 525393 D	01-01-1985
		ES 8502561 A	01-04-1985
		JP 59075304 A	28-04-1984