A parallel integrated frame synchronizer which implements a sequential pipeline process wherein serial data in the form of telemetry data or weather satellite data enters the synchronizer by means of a front-end subsystem and passes to a parallel correlator subsystem or a weather satellite data processing subsystem. When in a CCSDS mode, data from the parallel correlator subsystem passes through a window subsystem, then to a data alignment subsystem and then to a bit transition density (BTD)/cyclical redundancy check (CRC) decoding subsystem. Data from the BTD/CRC decoding subsystem or data from the weather satellite data processing subsystem is then fed to an output subsystem where it is output from a data output port.
Fig. 1

Fig. 2

Fig. 3

Fig. 4

Prior Art
Fig. 5
### Status Registers, 96

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Trailer Word/ Frmlength</td>
</tr>
<tr>
<td>0x04</td>
<td>Data Rate/ Misc Status</td>
</tr>
<tr>
<td>0x08</td>
<td>Timecode Status #0</td>
</tr>
<tr>
<td>0x0c</td>
<td>Timecode Status #1</td>
</tr>
<tr>
<td>0x10</td>
<td>Timecode Status #2</td>
</tr>
<tr>
<td>0x14</td>
<td>Timecode Status #3</td>
</tr>
<tr>
<td>0x18</td>
<td>Output Wenn 0</td>
</tr>
<tr>
<td>0x1c</td>
<td>Output Wenn 1</td>
</tr>
<tr>
<td>0x20</td>
<td>Check Frames</td>
</tr>
<tr>
<td>0x24</td>
<td>Lock Frames</td>
</tr>
<tr>
<td>0x28</td>
<td>Flywheel Frames</td>
</tr>
<tr>
<td>0x2c</td>
<td>Back-to-Search Events</td>
</tr>
<tr>
<td>0x30</td>
<td>CRC Error Frames</td>
</tr>
<tr>
<td>0x34</td>
<td>Inverted Frames</td>
</tr>
<tr>
<td>0x38</td>
<td>Frames with Sync Errors</td>
</tr>
<tr>
<td>0x3c</td>
<td>Frames with Slip Errors</td>
</tr>
<tr>
<td>0x40</td>
<td># Sync Errors</td>
</tr>
<tr>
<td>0x44</td>
<td># Slip Errors</td>
</tr>
</tbody>
</table>

### Set Up Registers, 94

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x48</td>
<td>General Chip Setup/Control</td>
</tr>
<tr>
<td>0x4c</td>
<td>Front-end Setup</td>
</tr>
<tr>
<td>0x50</td>
<td>Sync Pattern/Weather #1</td>
</tr>
<tr>
<td>0x54</td>
<td>Sync Pattern/Weather #2</td>
</tr>
<tr>
<td>0x58</td>
<td>Sync Mask/Weather #3</td>
</tr>
<tr>
<td>0x5c</td>
<td>Sync Mask/Weather #4</td>
</tr>
<tr>
<td>0x60</td>
<td>Correlator Setup/Weather #5</td>
</tr>
<tr>
<td>0x64</td>
<td>Frmlength Regs/Weather #6</td>
</tr>
<tr>
<td>0x68</td>
<td>Window Setup/Weather #7</td>
</tr>
<tr>
<td>0x6c</td>
<td>BTD/CRC/Weather #8</td>
</tr>
<tr>
<td>0x70</td>
<td>Output Setup</td>
</tr>
<tr>
<td>0x74</td>
<td>Timecode Regs #1</td>
</tr>
<tr>
<td>0x78</td>
<td>Timecode Regs #2</td>
</tr>
<tr>
<td>0x7c</td>
<td>Timecode Regs #3</td>
</tr>
</tbody>
</table>

**Fig. 7**
Fig. 8
Fig. 9
Fig. 10
Fig. 11
Fig. 12
Fig. 13
Fig. 14
Fig. 15
Fig. 16
Fig. 17
WEATHER SATELLITE DATA PROCESSING SUBSYSTEM

Fig. 18
Fig. 19A
Fig. 19B
Receive Data from Front-End Subsystem

Compare each byte of data with each byte of sync pattern for each bit position of the byte. Generate error count from the above comparison.

Add the error count for each bit position.

Compare the error count with the error tolerance.

Is error count = error tolerance?

Yes

True Sync

No

Is inverted error count = error tolerance?

Yes

Inverted Sync

No

Send data and Sync to Window Subsystem.

Fig. 20
Fig. 21A
Locks to sync marks separated by frame length

Is Run Mode set to Regular?

No

Locks to sync marks separated by frame length inclusively between two lengths

Is Run Mode set to Variable frame length?

No

Data is blocked up into frame length chunks.

Is Run Mode set to free-range?

No

Locks to sync marks determined by the first two sync marks

Is Best Match Strategy On?

Yes

If the first sync mark is seen but not the second, and another mark is seen, but the distance between the two sync marks is below the range, then employ the best match strategy to calculate the frame length.

Fig. 21B
Count the number of bytes between each sync marks

Accept the first sync and ignore this sync mark if another sync mark is found with fewer errors

Output the current frame as a short frame

Send the data onto the Data Alignment Subsystem

Continue to accept sync marks and do not look for another sync mark before the end of the frame
3

- Store the data onto FIFO
- Align the data to the byte boundary
- Invert the data (Yes or No)
  - Yes: Is data needed inversion?
    - Yes: Delete extra gained bits from the last byte of the frame. Add bits to the last byte of the frame if there was loss
    - No: Pad the frame with data to the frame length
  - No: Send the aligned data to the BTD/CRC Decoding Subsystem

Fig. 22
What is the BTD offset? i.e. at which byte number to start decoding? The offset value can be in the range of 1 to 8.

Use the polynomial \( h(x) = x^8 + x^4 + x^3 + x^2 + 1 \) to perform Bit Transition Decoding from the offset point to the end of the frame.

What is the CRC start and end offset? i.e. at which byte number to start calculate? The start offset value can be in the range of 1 to 8.

Use the polynomial \( g(x) = x^{16} + x^{12} + x^5 + 1 \) to compute the checksum from the offset point to the end point.

Compare the computed checksum against the transmitted checksum.

- Report the CRC error if not equal.
- No CRC error if equal.

Send the CCSDS Processed Data to Output Subsystem.
Duplicate the same byte of data on both byte lanes.  

Is data to be output in Byte mode?

Yes

Output the data in word mode

No

Generate two separate write enable for the data Generate Start of Frame (SOF) and End of Frame (EOF) marks

Append up to 8 bytes of frame status and 16 bytes of timecode to each frame

Output the data and other control signals on the output ports

Fig. 24
Get synchronized serial data from the Front-End Subsystem

Program the required polynomial in the PN sequence generators

Initialize the PN generators with incoming data stream

Compare the output of the PN generators with incoming data stream

Count the number of consecutive matches

Is that count equal to some programmed count?

Yes

No

Fig. 25A
Disconnect the PN generators from the incoming data stream and generated the rest of PN sequence.

Compare the output of the PN generators with the input data stream and store the number of mismatches.

Compare the 16-bit output word generated by PN generators with the stored sync pattern.

- Is it equal? (Is it equal?)
  - No
  - Is data PN encoded? (Is data PN encoded?)
    - No
    - Yes
  - Yes

Fig. 25B
Decode the data

Is every other word of data is complemented?

Correct the data

Get the word size and frame length from the setup registers

Are word size and frame length to be determined from the frame header?

Does frame header have triple redundancy?

Read the header once and determine the word size and frame size

Fig. 25C
Read the header three times and use the majority rule to determine the word size and frame size.

Format the input data stream into desired word size (word size can be in the range of 2 bits to 16 bits).

Is CRC Check needed?

- No
- Yes

Generate the CRC checksum and compare against the transmitted checksum.

Output the Weather processed data onto the Output Subsystem.

Output the Weather processed data status onto the Periodic and Cumulative Quality Counters Subsystems.

Fig. 25D
Start

Is TimeCode stamping needed?

Yes

Which Time Code i.e. External or Internal to use?

If Internal TimeCode

Load the start time

Use Ten Megahertz clock to count microsecond of the millisecond

Is count equal to 1000?

No

Append this timecode to each processed frame

Send an interrupt to indicate it is safe to write to internal timecode latches

Store up to 16 bytes of external timecode

If External TimeCode

No TimeCode will be appended to the frame

Append this timecode to each processed frame

Increment the MilliSecond of the DayCounter by 1

Fig. 26A
Prepend 8-bits of P-field to timecode if desired

Append 16 bits of daycount, 27 bits of millisecond of the day and 10 bits of microsecond of millisecond to each frame

Increment the DayCount by 1

Is count equal to 864000000?

No

Fig. 26B
Select the Status Source

Is this subsystem software controlled?

Output the contents of status registers every N number of frames where N is some number in the range of 1 frame to 64K frames

Output the contents of status registers on the periodic output port

Fig. 27
Count the number of frames with slip bits
Count the number of Flywheel frames
Count the number of Lock frames
Count the number of Check frames
Count the number of back-to-search events
Count the number of inverted frames
Count the number of frames with sync errors
Count the number of output mode controlled by SCLF
Count the number of sync errors in the data stream
Count the number of slip bits detected in the data stream
Count the number of frames with CRC errors

Store all the counts and make these counts accessible through microprocessor

Fig. 28
1
PARALLEL INTEGRATED FRAME SYNCHRONIZER CHIP
CROSS REFERENCE TO RELATED APPLICATION

This application now formalizes and incorporates herein by reference Provisional Application Ser. No. 60/028,733, "Parallel Integrated Frame Synchronizer Chip", Parminder S. Ghuman et al, filed on Oct. 15, 1996, and claims the priority date thereof under 35 U.S.C. §119(e).

ORIGIN OF THE INVENTION

The invention described herein was made by employees of the United States Government and by a contractor employee in the performance of work under a NASA contract and therefore is subject to Public Law 96-517 (35 U.S.C. §201 et. seq.) and may be manufactured and used by or for the Government for governmental purposes without the payment of any royalties thereon or therefor.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to apparatus for processing digital data and more particularly to a data frame synchronizer for return-link signal processing apparatus receiving a serial data stream from an airborne vehicle such as a spacecraft.

2. Description of Related Art

The rapid growth of chip fabrication densities has resulted in an extremely positive impact of telemetry data systems, particularly those of the National Aeronautics And Space Administration (NASA), the assignee of the present invention.

New data system implementations are currently becoming smaller, cheaper and more powerful due to the availability of higher integration components developed through improved very large scale integrated circuit (VLSI) fabrication processes. For ground telemetry systems, many of these components are the latest standard commercial microprocessors and solid-state memories developed for general purpose computing. Although general purpose components have improved telemetry data system implementations, even greater improvements are being obtained with the addition of components developed specifically for telemetry processing. This effort has led to a series of many different telemetry processing semiconductor components implemented in silicon and gallium arsenide. The high integration levels offered by these components have enabled the development of VLSI based systems that provide an order of magnitude improvement in performance cost and size heretofore unattainable.

To make full use of today's available VLSI densities, a new series of VLSI application specific integrated circuit (ASIC) components for return link data processing has been developed using the widely adopted packet telemetry protocols recommended, for example, by the Consultative Committee for Space Data Systems (CCSDS).

In the past, telemetry formats tended to be unique for each application or mission. This mission-unique development cycle has led to very high costs for the acquisition and to promote interoperability between ground processing facilities, space verification equipment, compatibility testing and launch support facilities. Return link processing takes place after the acquisition, demodulation and digitalization of signals transmitted from a spacecraft, e.g. a satellite, as shown in FIG. 1.

Return link processing systems generally extract framed digital data from incoming signal bit streams, correct frame to frame data, validate the protocol structures within the frame, and extract user data. A typical return link processing chain for packetized CCSDS telemetry signals is shown in FIG. 4 and includes apparatus for providing frame synchronization, Reed-Solomon error correction, and service processing.

While the present invention is primarily directed to frame synchronization of telemetry data, it also provides frame synchronization of data from a weather satellite.

Frame synchronization is the process of demarcation of telemetry frame boundaries from an incoming serial bit stream by use of attached sync markers as shown in FIG. 3.

Conventional CCSDS telemetry systems, moreover, use a specific digital sync marker pattern (ASM) such as shown in FIG. 3 to delineate data frame boundaries. Because space-to-ground transmission induces numerous types of data disturbances, frame synchronizers employ sophisticated measures in searching for these markers to ensure correct synchronization of data. Thereafter, Reed-Solomon error correction removes errors introduced during the transmission process while CCSDS processing demultiplexes, extracts and validates user data from the composite stream of telemetry frames.

Weather satellites transmit data in a pseudo noise (PN) pulse stream. Nevertheless, frame synchronization of such data is still required before any kind of data processing can be done.

SUMMARY

Accordingly, it is an object of the present invention to provide an improvement in apparatus which receives a digital data stream from airborne vehicules, such as satellites.

It is another object of the invention to provide an improvement in the return link signal processing of a serial data stream from satellites.

It is a further object of the invention to provide an improvement in apparatus for implementing data frame synchronization of the serial data stream input to return link signal processing apparatus.

The foregoing and other objects of the invention are achieved by the architecture of a parallel integrated frame synchronizer (PIFS) and its method of performing telemetry frame synchronization and its implementation on a semiconductor chip. It is controlled by a set of internal registers that are configured through a standard microprocessor interface prior to operation so as to meet the needs of a plurality of operational modes. The PIFS in accordance with this invention implements a sequential pipeline process wherein data enters the synchronizer by means of a front-end subsystem and passes to a parallel correlator subsystem or a weather satellite data processing subsystem. When in a CCSDS mode, data from the parallel correlator subsystem passes through a window subsystem, then to a data alignment subsystem and then to a bit transition density (BTD)/cyclical redundancy check (CRC) decoding subsystem. Data from the BTD/CRC decoding subsystem or data from the weather satellite data processing subsystem enters an output subsystem where it is fed to a data output port.
Further scope of applicability of the present invention will become apparent from the description provided hereinafter. It should be understood, however, that the detailed description and specific example set forth herein, while disclosing a preferred embodiment of the invention, is provided by way of illustration only, since various changes and modifications coming within the spirit and scope of the invention will become apparent to those skilled in the art with this detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will become more readily understood from the detailed description provided hereinafter and the accompanying drawings which are provided by way of illustration only, and thus are not meant to be limiting of the invention, and wherein:

FIG. 1 is a diagram illustrative of a typical down link telemetry system;
FIG. 2 is a diagram illustrative of the telemetry data consisting of a stream of fixed length transfer frames of data with attached sync markers (ASM) located therebetween;
FIG. 3 is a digital pattern illustrative of a typical 32-bit ASM;
FIG. 4 is a diagram illustrative of a typical return link processing chain for telemetry signals in accordance with the subject invention;
FIG. 5 is an electrical block diagram of a preferred embodiment of the parallel integrated frame synchronizer (PIFS) in accordance with the subject invention;
FIG. 6 is an electrical block diagram of a preferred embodiment of the parallel integrated frame synchronizer (PIFS) shown in FIG. 5;
FIG. 7 is a diagram identifying the set up registers and status registers shown in FIG. 6;
FIG. 8 is a block diagram illustrative of the front-end subsystem shown in FIG. 6;
FIG. 9 is an electrical block diagram illustrative of the parallel correlator subsystem shown in FIG. 6;
FIG. 10 is an electrical block diagram illustrative of the window subsystem shown in FIG. 6;
FIG. 11 is a search check lock flywheel (SCLF) state diagram illustrative of the algorithm implemented by the window subsystem shown in FIG. 10;
FIG. 12 is an electrical block diagram of the parallel integrated frame synchronizer (PIFS) shown in FIG. 5;
FIG. 13 is an electrical block diagram illustrative of the BTC/CRC decoding subsystem shown in FIG. 6;
FIG. 14 is an electrical block diagram illustrative of the output subsystem shown in FIG. 6;
FIG. 15 is an electrical block diagram illustrative of the timecode subsystem shown in FIG. 6;
FIG. 16 is an electrical block diagram illustrative of the periodic output subsystem shown in FIG. 6;
FIG. 17 is an electrical block diagram illustrative of the cumulative quality counters subsystem shown in FIG. 6;
FIG. 18 is an electrical block diagram illustrative of the weather satellite data processing subsystem shown in FIG. 6;
FIGS. 19A and 19B are flow charts illustrative of the operation of the front end subsystem shown in FIG. 8;
FIG. 20 is a flow chart illustrative of the operation of the parallel correlator subsystem shown in FIG. 9;
FIGS. 21A–21C are flow charts illustrative of the operation of the window subsystem shown in FIG. 10;
FIG. 22 is a flow chart illustrative of the operation of the data alignment subsystem shown in FIG. 12;
FIG. 23 is a flow chart illustrative of the operation of the BTC/CRC decoding subsystem shown in FIG. 13;
FIG. 24 is a flow chart illustrative of the operation of the output subsystem shown in FIG. 14;
FIGS. 25A–25D are flow charts illustrative of the weather satellite data processing subsystem;
FIGS. 26A and 26B are flow charts illustrative of the operation of the time code interface subsystem;
FIG. 27 is a flow chart illustrative of the operation of the periodic output subsystem shown in FIG. 16; and
FIG. 28 is a flow chart illustrative of the operation of the cumulative quality counter subsystem shown in FIG. 17.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to the drawings and more particularly to FIG. 1, shown thereat is an illustration of a typical telemetry system wherein telemetry data is transmitted from a spacecraft 10 to a ground telemetry processing facility 12 via a telemetry data relay spacecraft system (TDRSS) 14 in a modulated form as shown by reference numeral 16 where it is received by a parabolic dish antenna 18. The processing facility 12 demodulates the data and sends it to a frame synchronizer 20, which comprises the subject matter of this invention. The synchronizer 20 is required because space to ground communication induces errors in the down-link data stream 16.

Synchronization is achieved by locating valid sync marker patterns (ASM) 24 between fixed length data frames 26 in a serial data stream 22 so as to provide a composite frame of data 30 as shown in FIG. 2. The synchronizer 20 recognizes and corrects an ASM 24 (FIG. 2) in the demodulated data stream 22 that is used to delineate the beginning of each frame 26 of telemetry data with an ASM internally generated in the synchronizer 20 and allows error tolerance in the bit pattern of the incoming data stream 22. The ASM 24 for one transfer frame 26, immediately follows the end of the preceding transfer frame 26. There is no intervening bits between the ASM 24 and the transfer frame 26.

While not being limited thereto, the Consultative Committee for Space Data Systems (CCSDS) protocol, for example, typically calls for a 32-bit ASM having a pattern as shown in FIG. 3. The digital bit pattern shown in FIG. 3 can be represented in hexadecimal notation as 1ACFFC1d. However, a sync marker pattern up to 64 bits can be utilized when desired. The output of the frame synchronizer 20, as shown in FIG. 2, thereafter typically undergoes a well known Reed-Solomon error correction shown by reference numeral 31, and service processing as shown by reference numeral 32 before being sent to a data user 34 as shown in FIG. 4. The parallel integrated frame synchronizer (PIFS) 20 in accordance with the subject invention is implemented on a semiconductor chip mounted on a printed circuit board 36 as shown in FIG. 5.

Referring now to FIG. 5, associated with the PIFS chip 20 is a serial to parallel data converter 38, a synchronous first-in, first-out input (FIFO) memory 40 and a FIFO output memory 42. Operation of the PIFS 20 is controlled by an external microprocessor 44 and an external clock (CLOCK) signal from a system clock signal generator 46. The CLOCK signal is applied to the synchronizer 20 on signal line 47.

To input data into the PIFS 20, a user may select one of four different interfaces via a 32-bit bidirectional control bus
prises an asynchronous parallel data interface 48 coupled to the microprocessor 44. The first interface comprises a FIFO interface 50 which permits the user to input data (FIFO DATA) into the synchronizer 20 from the synchronous FIFO memory 40 via an 8-bit data bus 52 connected to an input port 54. The second interface comprises an asynchronous parallel data interface 56 which permits asynchronous high speed data (PDATA) to be coupled from an external asynchronous device, not shown, via the serial to parallel converter 38, and an 8-bit data bus 58 connected to an input port 60. The third interface comprises a serial data interface 62 which permits the user to input serial data (SDATA) into the PIFS 34 and comprises a 4-bit data bus 64 which connect to four corresponding serial data input ports shown by reference number 66 in FIG. 5. The fourth interface comprises a test interface 68 and which comprises an 8-bit data bus 70 from the microprocessor 44 to an input port 72 for inputting test data (CPU DATA) from a CPU, not shown, in the microprocessor 44 for "debugging" the PIFS chip 20.

A block diagram of the PIFS 20 is shown in FIG. 6. Referring now to FIG. 6, data passes through the PIFS 20 in a sequential pipeline process. Digital data including FIFO DATA, PDATA, SDATA and CPU DATA enters the chip via a front end subsystem 74 coupled to the four interfaces 50, 56, 62 and 68 shown in FIG. 5. The data out of the front end subsystem 74 passes either to a parallel correlator subsystem 76 or a weather satellite data processing subsystem 78. If the PIFS 20 is in the CCSDS down link mode wherein telemetry data other than weather data is received, data from the parallel correlator subsystem 76 passes to a window subsystem 80, then to a data alignment subsystem 82, and then to a bit transition density (BTD)/cyclical redundancy check (CRC) decoding subsystem 84. Data from the BTD/CRC decoding subsystem 84 or data from the weather satellite data processing subsystem 78 is then fed to an output subsystem 86.

The PIFS 20 also includes a timecode interface 90 which will be described subsequently. A control and status section 92 which is comprised of a plurality of setup registers 94, and a plurality of status registers 96 shown in the diagram of FIG. 7, a periodic output controller 98 and a plurality of cumulative quality counters 100 for providing housekeeping and data flow control.

Considering now each of the various subsystems shown in FIG. 6, the front end subsystem 74 is shown in greater detail in FIG. 8 and includes an internal input synchronizer 102, a rate counter 104, a reserializer 106, a timeout mechanism 108, a rate regulator 110, a timeout counter 112, and a front end clock 114. The internal input synchronizer 102 also includes a NRZ decoder 116 which is used to convert, if needed, input data into an NRZ-L format if the inputted data is coded in an NRZ-M or an NRZ-S format. The purpose of the front-end subsystem is primarily to synchronize the input data to a system clock i.e. the clock signal generator 46 (FIG. 1) and, where necessary, convert serial data into parallel data.

As shown in FIG. 8, the input synchronizer (INPUT SYNC) 102 receives as inputs four data signals and a clock signal and shown as: FIFO DATA on input data bus 52, PDATA on input data bus 58, SDATAS on input data bus 64, CPU DATA on input data bus 70, and a CLOCK signal on signal line 47. The INPUT SYNC 102 operates to synchronize the inputted data to the CLOCK signal on line 47 and convert serial data SDATA input on bus 64 to parallel data. If the inputted data needs decoding, it is applied to the NRZ decoder 116. If no decoding is necessary, the synchronized data is fed to a parallel data output bus 118. It is also fed to a rate counter (RATE-CNT) 104 or an output unit termed the front-end clock (FRONT-ENDCLK) 114. If decoding is necessary, the synchronized data is first fed out of the input synchronizer 102 on bus 120 where it is reserialized in the unit 106 before being fed to data bus 122 to the front-end clock (FRONT-ENDCLK) unit 114. The FRONT-ENDCLK 114 operates to output serial data (SERDATA) on the data bus 124 which is synchronized to the system CLOCK signal on signal lead 47, generates a sync write (SYNCWRT) signal on line 126, and a serial enable signal (SERENA) on signal line 128.

The rate counter (RATE-CNT) 104 in the front end subsystem 74 is responsive to the CLOCK signal on line 47 and the parallel data on bus 118 to determine the number of input data bytes received during a selectable number of system clock cycles and allows the user to monitor the data rate of the incoming data. The rate counter 104 outputs the data rate (DATARATE) fed to the rate register (RATE-REG) 110 via signal line 125 in response to a rate select (RATESEL) input control signal on line 127. The DATARATE output signal appears on signal line 129.

The front-end subsystem 74 also includes a user programmable timeout feature which operates to flush, i.e. feed forward data presently in the pipeline out to the next subsystem, i.e. the parallel correlator 76 (FIG. 9), if data is not being input for a predetermined number of system clock cycles. This is provided by a mechanism termed TIME-OUTSM and is responsive to data to be outputted from the INPUT SYNC 102 which appears on bus 130 and the CLOCK signal on line 47. The data in the pipeline to be flushed is ordered in response to a TIMEOUT control signal which is generated in the TIME-OUTSM 108 and which appears on signal line 132. The TIME-OUTCNT unit 112 comprises a counter which merely keeps track of how many bytes of data which can remain without a timeout signal being generated.

Referring now to FIG. 9, shown thereat is a block diagram of the parallel correlation subsystem 76. It is comprised of eight identical data bit comparators 134, 134, . . . 134, a shift comparator (SHIFTCOMP), eight digital adders 136, 136, . . . 136, eight digital signal generators 138, 138, . . . 138, and eight syndrome generator/ sync marker (SYNCGEN) and a sync marker output selector (SYNCSEL) 140.

The parallel correlator subsystem 76 compares the parallel data stream outputted from the front end system 74 via data bus 118 on a bit-by-bit basis to a stored sync marker pattern (SYNMARK) applied via bus 142 and which has initially been set up in the setup registers 20 or 21, shown in FIG. 7, under the control of the microprocessor 44, the purpose being to detect the presence of a valid sync marker or ASM 24 (FIG. 3) in the data stream. The SHIFTCOMP comparators 134, . . . 134, respectively receive successive single bits of an 8-bit data byte of the data stream DATAIN on bus 118 and correlates a sync marker pattern (ASM) of up to 64 bits in parallel fashion. Accordingly, the first SHIFTCOMP 134, accepts the first bit appearing on data bus 118, the second SHIFTCOMP 134, receives the next or second bit of the same data byte, and so on, with the eighth bit of the 8-bit data byte being fed to SHIFTCOMP 134.

Each respective bit is compared against a corresponding bit of an expected sync marker data byte which has been fed to a respective SHIFTCOMP 134, . . . 134, and any error
determined thereby is fed to a respective pipeline adder PIPEADD. The PIPEADDs $136, \ldots, 136_p$ calculate the total number of correlation bit errors detected. If the sync marker pattern (ASM) sought is comprised of 32 bits, 4 consecutive 8-bit data bytes are sequentially correlated in a single correlation cycle.

When desirable, some of the bits can be masked. This is accomplished by data bit (MASK) inputs being fed from the setup registers 94 (FIG. 7) via data bus 144. Once 4 bytes making up the 32-bit sync marker pattern are processed, the adders $136, \ldots, 136_b$ add up all of the bit mismatches. This number is then fed to the set of sync marker generators (SYNGEN) $138, \ldots, 138_b$, which determine whether a valid sync marker (ASM) is present based upon the total errors generated by the PIPEADD adders $136, \ldots, 136_b$. This is achieved by comparing the total errors counted against an error tolerance set by the user in the set up register 26 (FIG. 7). If the total number of errors in the four bytes of data are less than the designated error tolerance, a valid sync marker is generated by each of the sync generators (SYNGEN) $138, \ldots, 138_b$. The sync marker selector (SYNSEL) 140 accepts all of the sync markers generated by the SYNGENs and selects one of the sync markers which has at a bit position which coincides with the bit where the data started.

The eight sync generators (SYNGEN) $138, \ldots, 138_b$, in addition to calculating a valid sync marker based upon the total errors generated by the adders $136, \ldots, 136_b$, also determine whether the incoming data (DATAIN) is true or inverted data. The sync selector 140 then outputs the data on bus 146, outputs an error signal flag (ERROR) on line 148 if the DATAOUT on bus 146 comprises bad data, the generated sync marker (SYNC) on bus 150 and inverted data on bus 152 if the data is inverted. A flush (FLUSH) signal flag is generated on line 151 if the time out mechanism 108 in FIG. 8 generates a TIMEOUT signal on line 132, indicating the data presently in the pipeline is to be flushed out. After the parallel correlator subsystem 76 correlates the data and detects what appears to be a valid sync pattern (ASM) in the data stream, it passes the data (DATAOUT) to the window subsystem 80 shown in FIG. 10 for further validation. The purpose of the window subsystem 80 is to lock onto a valid sync marker when detected. Referring now to FIG. 10, the window subsystem 80 includes a check mode tolerance counter (CTOLCNT) 156 a flywheel mode tolerance counter (FTOLCNT) 158, a free frame comparator (FREEMCP) 160, a search/check/lock/flywheel state machine (SCLFSM) 162, a frame counter (FRMCNT) 164, a window register (WINDOWREG) 166, a slip tolerance selector (SLIPTOLSEL) 168, and a window generator (WINDOWGEN) 170. The window subsystem 80 determines whether or not a valid sync marker is being received under a predetermined set of error tolerances initially established by the set up registers 26 (FIG. 7), in accordance with an algorithm shown by the SCLF state diagram of FIG. 11 and which is implemented by the SCLFSM 162.

Initially a check tolerance count (CTOLCNT) signal is fed to the CTOLCNT counter 156 from the set-up registers #26 on lead 172. In a like manner, a flywheel tolerance count signal (FTOLCNT) is sent to the FTOLCNT counter 158 via the signal lead 174, along with two bit slip tolerance signals SLIPTOL0 and SLIPTOL1 which are fed to the slip tolerance select (SLIPTOLSEL) register 168 on circuit leads 176 and 178.

The window subsystem 80 operates in four different modes to lock on to a valid sync marker; a regular mode, a variable frame length mode, a free-range mode and a asynchronous block mode.
permit the window subsystem 80 to accept a sync marker that has slipped one or more bits or gained one or more bits within the programmed sync marker boundaries, indicating that the system should nevertheless accept this data. If slip tolerances SLIPTOLO and SLIPTOHI have been entered into SLIPTOSEL 168, the WINDOWGEN 170 will generate a window of tolerance around an acceptable sync marker. Accordingly, the window generator 170 outputs an indication of the slip count (SLIPCNT) on signal line 192, the slip direction (SLIPDIR) on line 194, and whether or not the slip (SLIPPED) is in the sync marker window on line 196.

Two different slip and error tolerances can be set. Accordingly, the valid values for slip tolerance are 0 to 4 bits. Slip tolerance allows the window subsystem 80 to recognize a valid sync marker by widening the window by the slip bits set in the set-up register 94 (FIG. 7). The error tolerance is set by the set-up registers 94 to tell the window subsystem how many bit errors should be allowed in an incoming sync marker to still be recognized as a valid sync marker. The range of error values permitted is from 0 to 31. Setting the error tolerance too “wide” will cause the window subsystem 62 to false lock randomly. Also, the check tolerance count (CTOLCNT) and the flywheel tolerance count (FTOLCNT) can be set from 0 to 15 frames. The check tolerance count tells the SCLFSM 162 how many frames to process in the “check” mode before proceeding to “lock” (FIG. 11). The flywheel tolerance count tells the window subsystem how many frames to process after it has fallen out of “lock” before it goes back into the “search” mode.

The scheme shown in FIG. 11 offers an optional “best match” strategy in the “check” or “lock” modes depending upon the check tolerance (CHKTOL) value. Best match strategy is implemented between the time the first sync marker is found and the subsequent sync marker is searched for within the designated frame length. After accepting the first sync marker, if there is another sync marker with fewer errors than the first sync marker, the subsequent sync marker is accepted and in such instance, the current frame is output as a short frame.

Turning attention now to the data alignment subsystem 82, reference will now be made to FIG. 12 which depicts the data alignment system 82 being comprised of a synchronous first-in, first-out alignment (SYNCFIFO ALIGN) memory 198, first and second boundary registers (BOUNDARY REGISTER) 200 and 202, an alignment finite state machine (ALIGNFSM) 204, an 8-bit byte regenerator (REGEN8) 206, an alignment register (ALIGNREG) 208, a multiplexer (MUX) 210, a data inverter (INVUR) 212, an alignment control signal generator designated (ALIGNNP) 214, and a 16-bit down counter (DCNT 16) 216.

The data alignment subsystem 82 takes DATA from the window subsystem 80 on bus 191 and aligns frame data 30 (sync marker ASM data 24 and transfer frame data 26) to specified byte boundaries. When slips occur which can add or subtract bits from the frame data 30, the last byte is filled with 1’s and 0’s. When a data flush occurs, the data frame is padded to frame length bytes and the boundary between two consecutive frames is saved as status that can be optionally appended.

As shown in FIG. 12, DATA, CLOCK and reset (RST) signals on bus 190 and signal lines 41 and 55 are fed into SYNCFIFO ALIGN 198. The type of operating mode (RUNMODE) in which the window subsystem 80 is operating is fed as a control signal to the alignment finite state machine (ALIGNFSM) 204 on signal line 205. The clock (CLK) and reset (RST) signals are also fed to the alignment register (ALIGNREG) 208. The clock (CLK) signal is also fed into the 16 bit down-counter (DCNT16) 216 along with a frame length (FRAMLEN) signal and, when required, an input test (TESTIN) signal on leads 207 and 209.

The frame data 30 (FIG. 2) which is stored in the SYNCFIFO ALIGN memory 198 is fed to the boundary registers 200 and 202 as well as the regenerator (REGEN8) 206. The boundary registers 200 and 202 generate a pair of outputs identified as boundary front (BOUNDFRNT) and boundary back (BOUNDBACK) which are fed to the following subsystem, i.e. the BTD/CRC decoding subsystem 84 shown in FIG. 13 on buses 203 and 205, telling it where data alignment of the frame data began and ended. The ALIGNFSM 204 takes the data frame input from memory 198 and aligns the data based upon the mode being run by the window subsystem 80.

The data in the first boundary register 200 is also fed therefrom to the alignment register (ALIGNREG) 208 along with data from REGEN8 206. ALIGNP 214 also gets the output from the REGEN8 206 and operates to generate a start of frame (SOF) signal on line 211 and a write enable (WEN) output signal on line 213, indicating that the DATAOUT from the inverter 212 on bus 215 is properly aligned and it comprises a valid block of frame data 30. The down counter (DCNT16) 216 merely counts frame lengths and keeps track of the frame as it is being processed by the ALIGNFSM 204.

Considering now the BTD/CRC decoding subsystem 84 (FIG. 6), the apparatus shown in FIG. 13 implements both optional bit density decoding (BTD), and cyclic redundancy check (CRC) error detection of the data stream outputted from the alignment subsystem 82 on bus 215.

The BTD/CRC decoding subsystem 84 includes A BTD clock input (BTDCLKIN) register 218, a BTD/CRC pipeline (BTDRCRCP0) block 220, a bit density (BTD) decoder 222, a cyclic redundancy check (CRC) decoder 224, a set of 8-bit registers (REGEN 8) 226, 228, 230 and 232, a CRC error (CRCCERR) comparator 234, a counter (LDCNT) block 236, and a control signal generator (LDGEN) 238.

Optional bit transition density decoding is performed on data streams 212 (FIG. 1) that have been BTD encoded at the transmitter. This encoding ensures a certain number of transitions in the data stream which helps the sync marker stay locked. A CCSDS recommended code (h(x)x^8+x^4+x^3+x+1) is used to perform BTD decoding.

The data input to the BTDCLKIN register 218 on bus 215 is BTD decoded in the BTD decoder 222 following transfer from registers 218, 226 and 228 and it is stored in the REGN8 register 230. The decoded data is then fed to the error comparator (CRCCERR) 234 along with the output of the CRC decoder 224. CRCCERR 234 performs the cyclic redundancy check (CRC) decoding with comparator 234 outputting a CRC error flag signal (CRCCERR) on line 240, indicating whether or not a CRC error exists. The LDCNT counter 236 operates to count the number of frames with BTD error while the signal generator LDGEN 238 generates a control signal for the counter 236 as well as the BTD decoder 222.

A BTD offset can be set from 0 to 8 bytes. This offset indicates the number of bytes to skip before decoding starts. For example, for a 32 bit sync marker, the input signal BTD OFFSET input on signal line 223 to the BTD decoder 222 is 0x4. This means that BTD decoding will begin on the fifth byte of the incoming data stream (DATA) on line 215 and will continue until the next start of frame of data is received.
CRC error detection performed in CRC decoder 224 is a method to detect errors in the data stream. The CCSDS recommended polynomial \( g(x) = x^{16} + x^{12} + x^5 + 1 \) and the inverse polynomial \( g'(x) = x^{32} + x^{28} + x^{24} + x^{20} + x^{16} + x^{12} + x^8 + x^4 + 1 \) are implemented in the CRC block 224. An initial value of the CRC polynomial \( g(x) \) is normally initially set, whereupon a “checksum” signal is computed on the data starting from an offset (CRCOFFSET) value fed to the CRC decoder 224 on line 225 and ending at 2 bytes minus the end value. This CHECKSUM is compared against the transmitted “CHECKSUM” in the data frame and an appropriate error bit is set and outputted as a CRC error signal on line 240 if the CHECKSUMs do not match. The frame data output DATAOUT from the BTDCRC decoding subsystem 84 is provided on bus 231 from REGEN8 register 232.

Similar to BTD, a programmable offset (CRCOFFSET) from 0 to 8 bytes is provided to indicate how many bytes to skip before starting to compute the CRC value. For example, to start computing the CHECKSUM immediately after a 32 bit sync mark, the offset is 0x4. The DATA OUT is shown appearing at the output of the 8 bit register 232 on bus 231. A SEARCH CHECK LOCK FLYWHEEL output is also output on line 241 from the BTDCRCPIP block 220.

Considering now the output subsystem 86 disclosed at FIG. 14, frame data 30 (FIG. 2) from the BTD/CRC subsystem 84 or data from the weather satellite data subsystem 78 is fed to the output subsystem 86, depending upon whether the synchronizer 20 is in the CCSDS mode or in the weather mode. However, the operation of this subsystem is the same, irrespective of which subsystem is providing data. The three main elements in the output subsystem 86 are a multiplexer (WMUX) 242, a trailer (TRAILER) 244 and a byte or word (BYTEWORD) output register 246. The output register 246 outputs data in the form of either bytes or words on a data output bus 248 along with an end of frame (EOF) signal on line 250, a start of frame (SOE) on line 252, with a write enable (PWEN) signal on lead 254, in response to outputs from TRAILER 244, an SCLF mask (SCFL/MASK) control signal on line 255 and an outmode (OUTMODE) control signal on line 257.

The multiplexer (WMUX) 242 is responsive to the input data DATAIN appearing on data bus 231, along with status signals for the sync (SYNCSTAT) and data (STADDATA) on lines 256 and 258. In the weather mode, the WMUX 242 receives, in addition to weather data on DATAIN bus 231, a data lock (WLOCK) signal on line 262, a start of frame (WSOF) signal on line 264 and write enable (WWEN) signal on line 266. The TRAILER block 244 permits up to 8 bytes of frame status or 16 bytes of timencode generated by the timecode subsystem 90 (FIG. 6) to be appended to each data frame in response to status enable (STATEN), data (TIMEDATA) and time enable (TIMELEN) control signals applied to the lines 268, 270 and 272. The TRAILER 244 also outputs status data quality (STATQUAL) information on line 274 along with a flush status (FLUSHSTAT) signal on line 276 which tells the user that the present frame of data being output is invalid.

The next subsystem that needs to be addressed is the timencode interface subsystem 90 shown in FIG. 6, and which is shown in further detail in FIG. 15. Two types of timecodes are provided by the subject invention, an internal timencode, and an external timencode. Referring now to FIG. 15, the timecode subsystem 90 consists of three counters 278, 280 and 282, which generate a microsecond of a millisecond (USOFMSC) count output on line 279, millisecond (MSOFDAYCT) count output on line 281, and a day count (DAYCNT) output on line 283. The counters 278, 280 and 282 count from a 10 MHz clock signal generated by the clock signal generator 46 shown in FIG. 5, and which is applied via clock signal line 47. The outputs of the three counters 278, 280 and 282 are respectively applied to one multiplexer (INTMUX) 284 of a pair of INTMUX multiplexers 284 and 286. The second multiplexer 286 receives a day count input from line 283 and an output from a time latch (TIMELAT) 288 which is adapted to receive a time interrupt signal from a time interrupt (TINTERR) block 290.

The internal timecode, which is CCSDS day segmented (CDS) is appended to the frame data via the TRAILER 244 (FIG. 14) when programmed via signal line 298 from INTMUX 284. CDS comprises six bytes that are always padded to byte boundaries. The CDS timecode includes 16 bits from the day counter (DAYCNT) 282, 27 bits from the millisecond of the day counter (MSOFDAYCT) 280, and 10 bits from the microsecond of millisecond counter (USOFMSC) 278. The starting values for these counters are inputted via microsecond of millisecond (USOFMSC) time load (TIMELOAD) and reset (RST) control signals entered from the microprocessor 44 on lines 292, 294 and 296, respectively.

The external timecode is user defined and can be up to 16 bytes and reads a predetermined amount of time from an external source, such as a microprocessor 44 shown in FIG. 5, and stores it into the time latch (TIMELAT) block 288. INTMUX 286 decides whether it is going to use the internal or external timecode per the output TIMEDATA signal which appears on output line 300. The INTMUX 284 is responsive to a PFIELD on line 30 command, and determines the type of format and what type of time resolution TIMEDATAS is to be outputted on line 298.

Referring now to FIG. 16, shown thereat are the details of the periodic output subsystem 98 which forms part of the control and status block 92 shown in FIG. 6. The subsystem 98 is comprised of a 256 multiplexer (MUX2XT01X16) 306, a 16 bit register (REGEN16) 308, a 28 bit counter (COUNT128) 310, a periodic state machine (PERIODICSM) 312 and a periodic counter (PERCOUNT) 314.

The purpose of the periodic output subsystem 100 is to output status information from the status registers 96 (FIG. 7). The multiplexer 306 receives eight information input signals including check frames (CHCKFRMS), CRC frames (CRCFRMS), data rate (DATARATE), (ENCSYNC), lock frames (LOCKFRMS), inverted frames (INVFRMS), pin bit count (PNBITCT), and trailer word status (STATWORD) on lines 316, 318 . . . 330. The multiplexer 306 cycles in a periodic sequence by the output of the counter (COUNT128) 310, which receives a clock (CLOCK) input signal and a reset (RST) signal along with a control signal outputted from the periodic state machine (PERIODICSM) 312. The PERIODICSM 312 determines the frequency of the status outputs in response to an operational mode input signal on line 332 and an output from the periodic counter (PERCOUNT) 314 via line 334. The periodic counter 314 also sends out a test control signal (TESTOUT) 336. The PERIODICSM 312 outputs a write enable (WEN), a first-in, first-out clock (FIFOCLK) signal, and a set up clear (SETUPCLR) signal which appear on lines 338, 340 and 342. The multiplexed status outputs (OUTDATA) are fed from the register (REGEN16) 308 on signal line 231.

The periodic output subsystem 98 essentially operates in two distinct modes. In one mode, the subsystem is software
controlled. In this mode, the contents of the status registers 96 (FIG. 7) are written out as OUTDATA to the WMUX 242 (FIG. 14) via bus 231 and operates independently of any data flow. In the second mode, the subsystem outputs status data on bus 231 every N frames, where N is some number in the range between one frame to 64 K frames.

Along with the periodic output subsystem 98 in the control and status section 92 (FIG. 6) is a set of 12 counters included in a cumulative quality counter subsystem 100.

As shown in FIG. 17, the cumulative quality counter subsystem 100 is comprised of a control block (QOULSES) 346 and 12 counters 348, . . . 348, which operate in response to commands outputs from QOULSES 346 to output the number of counted lock frames, check frames, flywheel frames, frames with sync errors, frames with slip bits, frames with CRC errors, inverted frames, back-to-search events, sync errors and slip bits. The QOULSES 346 requests to respond to the microprocessor 14 (FIG. 5) via seven input signals shown as: back-to-search (BACK2Srch), (CRC ERROR), inverted frame count enable (IFRMCFTEN), inverted frames (INVERTFRM), search check lock flywheel (SCLF), serial frame error (SERROR) and end of frame (WEOF) which appear on signal lines 352, 354, . . . 364.

Basically, the cumulative quality counter interface 100 and the periodic output control 98 implement housekeeping functions for the PIFS chip 20. The interface from the external microprocessor 44 (FIG. 5) comprises the 8 bit bus 70 and the 32 bit bus 48 and allows the user to set/reset operational parameters and read status parameters to control/ determine the behavior of the PIFS 20.

Turning attention now to the weather satellite data processing (WSDP) subsystem 78 shown in FIG. 6, a block diagram of this subsystem is shown in FIG. 18. The WSDP 78 is used to process weather satellite data, as opposed to telemetry data, for several satellite systems, including, for example, DMSP and TIROS, GMS GOES-GVAR, and METEOSAT.

As shown in FIG. 18, the WSDP subsystem 78 includes a serial data correlator 366 including four pseudo noise (PN) generators (PN-A) 368, (PN-Al) 370, (PN-B) 372 and (PN-B1) 374. The correlator compares the respective PN sequence generated by the PN generators 368, 370, 372 and 374 against an incoming serial weather data stream DATA appearing on line 376. The four pseudo noise (PN) sequences have maximum length of 64 K bits, with two of the four PN sequences being the inverse (I) of the first two sequences A and B, so that two different PN sequences and their inverted form can be correlated. The pseudo noise generators 368, 370, 372 and 374 in effect implement a polynomial corresponding to the same pseudo noise pulse stream that is being generated by the weather satellite.

Once the correlator 366 finds the start bit of the weather input data (DATA), the correlated serial data enters a decode/word complement (PN-DECODER/WORDCOMPL) block 382 which decodes and complements the correlated data. Once the correlated data is decoded and complemented, if necessary, the data passes into a HEADER DATA EXTRACTOR 384 as well as a DATA ALIGNMENT block 386. The header is read three times and a majority decision is made in the data extractor 384 to decode the word length and the frame length of the incoming data.

The data alignment block 386 packs the serial data stream into a desired word size which can be in the range of two bits per word to 16 bits per word, with the data being output as bytes regardless of word size on data bus 388. If the word size is less than 8 bits, then the uppermost bits of the byte are filled with 1's or 0's. The data processed in the data alignment block 386 is also passed to a CRC DETECTION block 390. Again, the detector 390 detects error in the data stream; however, a different polynomial is set and a CRC CHECKSUM is computed on the data, starting from the first bit of the data, and ending at 2 bytes minus the 12 bytes of a particular block of data.

CHECKSUM is then compared against the transmitted CHECKSUM and an appropriate error bit is set in the status field via line 392 if the CHECKSUMs do not match. A word enable (WEN) signal is also generated by the data alignment block 386 and appears on output line 394. Two status registers 396 and 398 are also included in the WSDP subsystem 78, one being coupled to the correlator 366 and one being coupled to the header data extractor 384. Status output signals appear respectively on signal leads 400 and 402. The data output from the weather satellite data processing subsystem 78 is fed via bus 388 to the input bus 231 to the output subsystem 86, shown in FIG. 14, so that the output subsystem outputs either weather data or telemetry data, which can be used as valid data suitable for processing, for example, by Reed-Solomon error correction device 30 shown in FIG. 4, which is then fed to a data user, as shown in FIG. 1.

Referring now to the data flow charts shown in FIGS. 19–29, they are provided to further illustrate the flow of data through the various subsystems shown and described above.

FIGS. 19A and 19B comprise a flow chart for the front end subsystem 74. As shown, the first step in performing synchronization in accordance with this invention is to select the data source from the PDATA input port, the SDATA input port, CPU DATA input port or the FIFO DATA input port as shown by reference numeral 410. If the input data is serial data or has been serialized, it is first synchronized to the system clock as shown by step 412. The synchronized serial data is then converted to parallel data shown by reference numeral 414 and again synchronized to the system clock at step 418. The synchronized parallel data is then sent to the parallel correlator subsystem 76 shown by reference numeral 20. Next the number of parallel bytes of data are counted and stored, shown by reference numerals 422 and 424. The synchronized parallel data is next subjected to a TIMEOUT determination as shown by reference numeral 426 in FIG. 19B. If a TIMEOUT sequence has been initiated and the clock count exceeds a set count per step 428 and 430, a TIMEOUT and FLUSH command is generated as shown by reference numeral 432.

If on the other hand the input data is parallel data or CPU data, it is sent to the parallel correlator subsystem per step 420 unless it is NRZ-M/S decoded, in which case it is decoded first as shown by step 419 before being sent to the parallel correlator subsystem.

Next, the flow chart for the parallel correlator subsystem 76 shown in FIG. 9 is depicted in FIG. 20. As shown, parallel data is received from the front end subsystem 74 in 8-bit bytes via bus 118 (FIG. 9) per step 434, wherein each byte of input data is compared with each byte of the sought after sync marker pattern for each bit position 1 through 8 of the byte and an error count is generated per step 436. The error count for each bit position is added in the PIPEADD 136, 136a, shown by reference numeral 438, and compared with a preset error tolerance per step 440. If the error count is equal to the tolerance, a true sync marker pattern is indicated as being received per step 442, where-
Upon the data and the sync marker pattern are sent to the window subsystem \( \text{step } 444 \). If the error count is not equal to the error tolerance, a second query \( \text{step } 445 \) is made to determine if the inverter error count is equal to the error tolerance. If the answer is negative, the comparison is repeated. If the inverter error count is equal to the tolerance, the inverted sync marker is inverted per step \( \text{step } 446 \) and sent to the window subsystem as shown.

Considering now the flow chart for the window subsystem, reference is now made to Figs. 21A–21C and uses the synchronization strategy shown in the search-check-lock-flywheel (SCLF) state diagram shown in Fig. 11. As shown in Fig. 21A, a bit slip tolerance is first selected, as shown by reference numeral \( 448 \). If the position of the sync marker is less than 4 bits, 8 bits are added to the upper and lower bits of the sync marker plus a slip tolerance as shown by reference numeral \( 449 \). If the sync marker position is greater than 4 bits, then slip tolerance bits are added to the upper and lower bits of the sync marker position per step \( 450 \). If the new sync marker position is equal to the current sync marker position, then there is an indication that there is no bit slip as evidenced by reference numeral \( 451 \). If the new sync position is more than the current sync position, then the bit slip gain is noted per step \( 454 \). If the new sync position is less than the current sync position, then the bit slip loss is noted per step \( 456 \).

Moving on to Fig. 21B, if the RUNMODE is set to the regular mode, a lock mode is entered into where the sync markers are separated by the frame length per step \( 458 \) (Fig. 21B). If the RUNMODE is set to the frame length mode, then the sync markers are locked by the frame length designated per step \( 460 \). If the run mode is set to a free range mode, then the data is blocked into frame length chunks. If the best match strategy is not employed, locking to sync markers is determined by the first two sync marks per step \( 464 \). If the best match strategy is employed, if the first sync marker is seen but not a second, and a subsequent mark is seen, but the distance between the two sync marks is below a predetermined range, then the best match strategy is used to calculate the frame length per step \( 466 \).

With reference to Fig. 21C, the window subsystem flow chart parallels after step \( 456 \) shown in Fig. 21A, whereupon the number of bytes between each of the sync markers is counted and outputted per steps \( 468 \) and \( 470 \). If the best match strategy is not employed, sync markers continue to be accepted and another sync marker pattern is not looked for before the end of a data frame per step \( 472 \). If the best match strategy is ON, the first sync marker is accepted, except if another sync marker is found with fewer errors, as indicated by reference numeral \( 474 \). The current frame is outputted as a short frame if necessary per step \( 476 \) and the data is sent to the data alignment subsystem \( 482 \) shown in Fig. 12 as shown by reference numeral \( 478 \).

The flow chart for the data alignment subsystem \( 78 \) is shown in Fig. 22 where the data inputted from the data alignment subsystem is first stored on a first-in, first-out (FIFO) basis in the memory \( 198 \) (Fig. 12) as shown by reference numeral \( 480 \). Next, the data is aligned to the byte boundaries of a data frame as indicated by reference numeral \( 490 \), and the data is inverted, if need be, per step \( 492 \). If the current data frame constitutes a slip frame, then the extra gain bits are deleted from the last byte of the frame. However, bits are added to the last byte of the frame if there was a loss per step \( 494 \). If the data frame constitutes a FLUSH frame, the frame is padded with data to the required frame length as shown by reference numeral \( 496 \), otherwise the aligned data is sent to the BTD/CRC decoding subsystem \( 84 \) shown in Fig. 13, per step \( 498 \).

Referring now to Fig. 23, shown thereat is the flow chart for the BTD/CRC decoding subsystem of Fig. 13 which is first evidenced by a reception of the aligned data from the data alignment subsystem \( 82 \) per step \( 500 \). If the incoming data stream was not optional bit transition density (BTD) encoded at the transmitter, it is immediately forwarded on to the output subsystem \( 86 \), shown in Fig. 14, per step \( 502 \). If no cyclic redundancy check (CRC) is needed, it is also sent to the output subsystem. On the other hand, in the event that BTD decoding is necessary, then the BTD offset is determined at step \( 504 \) and the CCSDS polynomial \( h(x) \) is used to perform bit transition decoding from the offset point to the end of the data frame, as shown by reference numeral \( 506 \). This data is then sent to the output subsystem \( 86 \) (Fig. 14).

As noted above, determination is also made as to whether or not CRC is needed. If it is needed, then a determination is made as to which byte to start error correction shown by step \( 508 \), using the recommended CCSDS polynomial \( g(x) \) wherein the CHECKSUM is computed from the offset point to the end point as shown by reference numeral \( 510 \), after which the computed CHECKSUM is compared against the transmitted CHECKSUM per step \( 512 \). The CRC error is then reported following the comparison shown by step \( 514 \), or else no CRC error is reported per step \( 516 \).

Next, the flow chart of the output subsystem is illustrated in Fig. 24. As shown, either CCSDS processed data or weather processed data is fed to the output subsystem \( 86 \) shown in Fig. 14 per step \( 518 \). If data is to be output in the “byte mode”, the same byte of data is duplicated on both byte lanes(?) as shown by reference numeral \( 520 \), whereupon two separate write enable (WEN) signals for data are generated along with a start of frame (SOF) and end of frame (EOF) control signals as shown by reference numeral \( 522 \). The same step is entered into if the output data is to be output in the “word mode” as shown by reference numeral \( 524 \). Next, up to 8 bytes of frame status and 16 bytes of timecode are appended to each data frame, as indicated by step \( 526 \), whereupon the data and other control signals are output on the output ports of the output subsystem \( 86 \) as indicated by step \( 528 \).
formatted into the desired word size, which can be in the range between 2 bits and 16 bits, as shown by reference numeral 548 shown in FIG. 25D after which a decision is made to see if a CRC check is required at 549. If no CRC check is needed, the formatted data stream is outputted to the output system 86 shown in FIG. 14, as indicated by step 552 and then the weather processed data status is output onto the periodic and cumulative quality counter subsystem 98 and 100 shown in FIGS. 17 and 16 in accordance with step 554. If a CRC check is needed, a CRC CHECKSUM is generated and compared against the transmitted CHECKSUM as shown in step 550 before outputting the weather processed data per step 552.

Returning to FIG. 25C, if every other word of decoded data provided by step 546 is complemented, then the data is corrected as shown by reference numeral 556 in FIG. 25C, whereupon word size and frame length are obtained from the set up registers 94 (FIG. 7) if necessary, in accordance with step 558. Otherwise, if the word size and frame length is to be determined from the frame header, then in accordance with step 559, Otherwise, if the word size and frame length is determined by 1 per step 576. If the count is not equal to a thousand, then the millisecond counter of the day is incremented by 1 as shown by step 578. If such a count does not occur, then the millisecond of the day counter is incremented by 1 again until such count is equal to the count necessary to increment the day counter shown by step 576 (FIG. 26A). Following this, 16 bits of “daycount”, 27 bits of millisecond of the day and 10 bits of microsecond of millisecond are appended to each data frame, as shown by reference numeral 580. Also, if desirable, 8 bits of P-field is prepended to the timecode as shown by step 582.

Referring now to FIG. 27, shown thereat is the flow chart for the periodic output controller subsystem 98 shown in FIG. 16. Inasmuch as the periodic output controller subsystem 98 monitors the status of either CCSDS processed data or weather processed data, in accordance with reference numerals 584 and 586, the source is selected per step 588 and if the periodic subsystem is software controlled, the contents of certain status registers 96 shown in FIG. 7, are output on the periodic output port coupled to the OUT-DATABUS 344 shown in FIG. 16 in accordance with step 590. If the periodic output subsystem 98 is not software controlled, then the contents of the status registers are output on OUTDATABUS 344 every N number of frames, where N is a number in the range between 1 and 64 K frames. This is indicated by step 592.

Next and finally, the flow chart for the cumulative quality counters 100 (FIG. 17) is depicted at FIG. 28. The flow chart of FIG. 28 simply shows that 12 different parameters are counted and stored and made accessible through operation of the microprocessor 44 shown in FIG. 5 in accordance with step 594 shown in FIG. 28. Reference numeral 596 indicates that the number of frames with slip bits are counted. Reference numeral 598 indicates that the number of flywheel frames are counted. Reference numeral 600 indicates that the number of lock frames are counted. In a like manner, the number of check frames are counted as shown by step 602. The number of back-to-search events are counted as shown by reference numeral 604. The number of inverted frames are also counted shown by reference numeral 606. The number of frames with sync errors are counted as shown by step 608. The number of output modes controlled by SCLF strategy shown in FIG. 11 is also counted per step 610. Also, the number of sync errors in the data stream are additionally counted as shown by step 614. The number of frames with CRC errors are also counted in accordance with the step shown by reference numeral 616. This concludes a discussion of the various flow charts for the sub-systems shown in FIG. 6.

Accordingly, what has been shown and described is a parallel frame synchronizer which is operable to: (a) provide rate detection with 24-bit rate counter and four levels of counter resolution; (b) performs correlation to any bit pattern up to 64 bits; (c) provides the capability to program bit error tolerance allowing up to 15-bit errors for correlation; (d) provides the capability to program frame size (up to 64 K bytes) and frame sync marker size; (e) provide the capability to program search, check, lock, and flywheel strategy (up to 15 check frames and up to 15 flywheel frames and an optional best-matched strategy); (f) perform slip correction (up to ±1 bit); (g) provide four modes of operation which include, fixed frame length, variable frame length, and asynchronous blocking of data; (h) provide optional cyclical redundancy check (CRC) decoding using Consultative Committee for Space Data Systems (CCSDS) recommended polynomial; (i) perform bit transition density (BTD) decoding using the CCSDS recommended polynomial; (j) provide programmable offset (0 to 8 bytes) for both BTD and CRC to indicating starting point; (k) provide the capability to perform time-stamping via an internally generated timecode (CCSDS day segmented) or an external 8-bit port; (l) output data and bytes or words; (m) allowing up to 8 bytes of frame status and 16 bytes of timecode to appended to frames; and (n) perform frame synchronization of weather satellite data received from various types of weather satellites.

Having thus shown and described what is at present considered to be the preferred embodiment of the invention and its method of operation, it should be noted that the same has been made by way of illustration and not limitation. Accordingly, all modifications, alterations and changes coming within the spirit and scope of the invention are herein meant to be included.
What is claimed is:

1. A parallel integrated data frame synchronizer for locating valid sync marker bit patterns between frames of data in a serial telemetry data stream and receiving clock signals from a system clock, comprising:
   (a) a front end subsystem for receiving synchronous and asynchronous input data from a plurality of input interfaces including parallel and serial interfaces;
   (b) a parallel correlation subsystem coupled to the front end subsystem and including a plurality of multi-bit data comparators for respectively determining, in parallel, an error for each successive bit position of a multi-bit data byte in the input data by comparing the byte against a data byte of an expected sync marker bit pattern, and a corresponding plurality of sync marker generators for respectively generating a valid sync marker pattern when the total number of bit errors resulting from comparing are within a predetermined error tolerance;
   (c) a window subsystem coupled to the parallel correlation subsystem for locking onto one of said valid sync marker bit patterns;
   (d) a data alignment subsystem coupled to the window subsystem for aligning bytes of data into data frames having predetermined boundaries following locking onto said one valid sync marker bit pattern;
   (e) a BTDiCRC subsystem coupled to the data alignment subsystem for selectively performing optional bit transition density (BTD) decoding on the serial data stream when so encoded to assist in sync marker bit pattern locking or performing cyclic redundancy check (CRC) error detection on the serial data stream for detecting errors in the serial data stream; and
   (f) an output subsystem coupled to the BTDiCRC subsystem for outputting the serial data stream for external use in byte or word format.

2. A parallel integrated frame synchronizer according to claim 1 wherein said front end subsystem includes: (a) a first data input port for receiving data from a synchronous FIFO interface synchronized with the system clock; (b) a second data input port for receiving asynchronous input data from an asynchronous parallel data interface; (c) at least one third serial data input port and at least one corresponding serial data input port for inputting data from an external microprocessor for testing purposes.

3. A parallel integrated frame synchronizer according to claim 1 wherein said front end subsystem includes means for synchronizing asynchronous input data to the system clock.

4. A parallel integrated data frame synchronizer according to claim 1 wherein said front end subsystem includes means for detecting the number of data bytes received during a selectable number of system clock cycles of the system clock.

5. A parallel integrated data frame synchronizer according to claim 1 wherein said front end subsystem includes timeout means for flushing data when input data is not received for a predetermined number of system clock cycles of the system clock.

6. A parallel integrated frame synchronizer according to claim 1 wherein said data alignment subsystem includes circuitry for padding the data frame to frame length bytes when flushing is called for by said front end timeout means.

7. A parallel integrated frame synchronizer according to claim 1 wherein front end subsystem includes a non-return to zero (NRZ) decoder for converting input data encoded in a NRZ-M or NRZ-S format into a NRZ-I format.

8. A parallel integrated data frame synchronizer according to claim 1 wherein said parallel correlation subsystem additionally includes a sync marker selector for selecting a valid sync marker bit pattern generated by said plurality of sync marker generators and which has a bit position which coincides with the bit where the input data started.

9. A parallel integrated data frame synchronizer according to claim 1 wherein said parallel correlation subsystem additionally includes a plurality of adders equal in number to the number of said data comparators for determining the total number bit errors in the respective data bytes of the input data.

10. A parallel integrated frame synchronizer according to claim 1 wherein said sync marker generators in said parallel correlation subsystem also determine whether the input data is true or inverted data.

11. A parallel integrated frame synchronizer according to claim 1 wherein said window subsystem is operable to lock on said valid sync marker in four different modes including:
   (a) a regular mode wherein valid sync marker bit patterns are separated by predetermined constant frame data length bytes programmable by a user; (b) a variable frame length mode wherein said valid sync marker bit patterns are separated by at least two user programmable data frame lengths; (c) a free-range mode wherein said valid sync marker patterns lock to a data frame in accordance with a predetermined slip error tolerance and error tolerance testing algorithm; and (d) an asynchronous block mode wherein the first byte of data is tagged as a first data frame and the data is thereafter blocked into predetermined frame length blocks.

12. A parallel integrated frame synchronizer according to claim 1 wherein said algorithm comprises an algorithm which effects a best match strategy between a valid sync marker bit pattern and a designated data frame length.

13. A parallel integrated frame synchronizer according to claim 1 wherein said algorithm comprises a search/check/lock/ilywheel algorithm.

14. A parallel integrated frame synchronizer according to claim 1 wherein said front end subsystem includes means for inverting said data frames when parallel correlation subsystem determines the input data to be inverted data.

15. A parallel integrated frame synchronizer according to claim 1 wherein said data alignment subsystem includes means for filling the last byte of a data frame when slips occur which can add or subtract bits from the data frame.

16. A parallel integrated frame synchronizer according to claim 1 wherein a CCSDS code as represented by \((h(x)=x^8+x^7+x^3+1)\) is used in performing said BTD decoding in said BTDiCRC subsystem.

17. A parallel integrated frame synchronizer according to claim 1 wherein said BTDiCRC subsystem includes means for skipping a predetermined number of bytes of data before performing BTD decoding.

18. A parallel integrated frame synchronizer according to claim 1 wherein a CCSDS cyclic redundancy code as represented by \(g(x)=x^{12}+x^{11}+x^6+1\) and its inverse \((g(x))^{-1}=x^{12}+x^{11}+x^6+1\) are used in performing CRC error detection in said BTDiCRC subsystem.

19. A parallel integrated frame synchronizer according to claim 1 and further comprising:
   (g) a timecode interface subsystem including means for appending an internally generated timecode or an externally generated timecode to a data frame output from the output subsystem.
21. A parallel integrated frame synchronizer according to claim 19 wherein said internally generated timecode is day segmented.

22. A parallel integrated frame synchronizer according to claim 19 wherein said externally generated timecode is user defined.

23. A parallel integrated frame synchronizer according to claim 1 and further comprising:

(h) a control and status section including a periodic output controller subsystem, a cumulative quality counter subsystem, a plurality of set-up registers and a plurality of status registers.

24. A parallel integrated frame synchronizer according to claim 22 wherein said periodic output subsystem is coupled to the group of status registers for periodically outputting the contents of the status registers to an output port in one of two modes, said modes including a mode wherein said output is output on a periodic basis independent of any data flow and a mode wherein output is output every N data frames, where N≧1.

25. A parallel integrated frame synchronizer according to claim 1 and further comprising:

(i) a weather satellite data processing subsystem for processing serial input weather data transmitted from a weather satellite system and feeding processed weather data to said output subsystem.

26. A parallel integrated frame synchronizer according to claim 25 and wherein said weather satellite data processing system includes a serial data correlator including a plurality of pseudo noise generators generating respective pseudo noise sequences for correlating the serial input weather data against said pseudo noise sequences and generating a correlated serial data stream and determining a start bit of the serial data stream; a pseudo-noise decoder for decoding the correlated serial data stream in a predetermined word size and mining the word length and data frame length of the decoded serial data stream; a header data extractor and a data alignment block for aligning bytes of data into data frames having predetermined frame length; a parallel integrated frame synchronizer according to claim 22 wherein said cumulative quality subsystem selectively performing optional bit transition density (BTD) decoding on the serial data stream when so encoded to assist in sync marker bit pattern locking or performing cyclic redundancy check (CRC) error detection on the serial data stream for detecting errors in the serial data stream; and

(f) outputting the serial data stream for external use in byte or word format.

27. A method in accordance with claim 29 wherein said plurality of interfaces include: (i) a synchronous FIFO interface; (ii) an asynchronous parallel data interface; (iii) a serial interface; and (iv) a test interface for inputting test data.

28. A method in accordance with claim 29 wherein the correlating step (b) includes determining the total number bit errors in the respective data bytes of the input data.

29. A method in accordance with claim 29 wherein said locking step (c) comprises locking on said one valid sync marker by way of four different locking modes including: (i) a regular mode wherein said valid sync marker bit patterns are separated by predetermined constant data frame length bytes programmable by a user; (ii) a variable frame length mode wherein said valid sync marker bit patterns are separated by at least two user programmable data frame lengths; (iii) a free-range mode wherein said valid sync marker patterns lock to a data frame in accordance with a predetermined slip and error tolerance testing algorithm; and (iv) an asynchronous block mode wherein the first byte of data is tagged as a first data frame and the data is thereafter blocked in into predetermined frame length blocks.

30. A method in accordance with claim 33 wherein said algorithm comprises a search/check/lock/flywheel algorithm.

31. A method in accordance with claim 33 wherein said algorithm comprises an algorithm which effects a best match strategy between a valid sync marker bit pattern and a designated data frame length.

32. A method in accordance with claim 29 wherein said decoding step (e) includes optional bit transition density (BTD) decoding using a CCSDS code as represented by (h(x)=x^8+x^4+x^3+x+1).

33. A method in accordance with claim 29 wherein said decoding step (e) of performing cyclic redundancy check (CRC) error detection includes using a CCSDS cyclic redundancy code as represented by (g(x)=x^16+x^12+x^5+1) and its inverse (g(x))^-1=x^16+x^12+x^5+1).

34. A method in accordance with claim 29 and further comprising the step of:

(g) appending an internally generated timecode or an externally generated timecode to an output data frame.
39. A method in accordance with claim 29 and further comprising the step of:

(h) processing a serial data stream consisting of weather data transmitted from a weather satellite system.

40. A method in accordance with claim 39 wherein said step (h) of processing weather data includes: generating a plurality of pseudo noise sequences, correlating the serial data stream consisting of weather data against said pseudo noise sequences, generating a correlated serial data stream, determining a start bit of the data stream, decoding the correlated data stream, generating a complement of the correlated decoded serial data stream, determining the word length and data frame length of the decoded serial data stream, formatting the decoded serial data stream in a predetermined word size, and feeding the serial data stream to an output port as bytes regardless of word size.