Data Decoder Assembly

C. R. Grauling
DSIF Digital Systems Development Section

Future deep space missions (e.g., Pioneer F/G) will be using convolutional coding. The present configuration of the Deep Space Network (DSN) is not suited to perform the decoding of this class of codes. This function (amongst others) will be performed by the Data Decoder Assembly which is scheduled for installation in the DSN in September 1971. This article presents a description of the Data Decoder Assembly and its implementation.

I. Introduction

The Data Decoder Assembly (DDA) is a new addition to the DSIF Telemetry and Command Subsystem scheduled for installation in the DSN in September 1971. The DDA will be capable of performing three mutually exclusive functions: sequential decoding of convolutionally encoded data, block decoding of 32/6 or 16/5 biorthogonal block coded data, and high-rate data formatting of coded or uncoded data for transmission on the Wideband Data Link with simultaneous recording of the data on magnetic tape. The sequential decoder function will be implemented at approximately 25,000 computations per second and will be useful at data rates of up to 2048 bps. The block decoding function will be used at the 26-m antenna sites only and will be capable of decoding at data rates of up to 2048 bps. The high-rate data formatting function is required at the 64-m antenna sites only and will be implemented at rates of up to 250 kbps.

This article presents a description of the implementation of the DDA. The major DDA component is a small microprogrammable digital computer. The discussion is in three parts. Each part is a description of the hardware, firmware, and software development, respectively.

II. DDA Hardware

The DDA consists of a single standard DSIF equipment rack. Each rack contains the following equipment:

(1) DDA Central Processing Unit (CPU)—Interdata Model 4 computer.

(2) Interface electronics assembly.

(3) Power supplies.

(4) HSDL/WBDL buffer (at 64-m antenna sites only).

Figure 1 is a block diagram of the DDA. The DDA Central Processing Unit and interface electronics assemblies are briefly described below.

A. DDA CPU

The DDA CPU is an Interdata Model 4 computer with the following optional equipment:

(1) Two high-speed direct memory channels (selector channels).

(2) Magnetic tape controller and selector channel (at 64-m antenna sites only).

(3) Sixteen-line interrupt module.

(4) Four 16-bit programmable input/output (I/O) channels.

The computer is microprogrammable and has a full instruction set which is an emulation of a subset of the IBM 360/20 instruction set. In addition there is a set of special DDA instructions which are used to implement functions in which computation speed is critical, such as the sequential decode function.

B. Interface Electronics Assembly

The interface electronics assembly consists of a set of functional subassemblies. Each functional subassembly consists of a single IC socket panel with wirewrap interconnections. Each socket panel contains from 100 to 150 sixteen-pin dual in-line package integrated circuits. The socket panels plug into a wirewrapped backplane assembly which accounts for all the interconnections between subassemblies and external equipment. Each functional subassembly is briefly described below.

1. FTS/DDA coupler. This subassembly provides a means by which the DDA CPU can obtain Greenwich Mean Time (GMT) from the Frequency and Timing System (FTS). GMT is always available in binary-coded decimal (BCD) format via this coupler. This subassembly also contains a millisecond counter and hundredths of a second counter. The millisecond counter clears at one-second intervals and is readable by the DDA CPU. The hundredths of a second counter automatically clears at midnight GMT and is both readable and loadable by the DDA CPU. This coupler also generates three interrupts synchronous with the 1-kpps, 100-pps, and 1-pps signals which are available from the FTS.

2. SSA/DDA coupler. This functional subassembly provides the interface necessary to allow symbols to be transferred directly into the DDA CPU core memory via a selector channel. Some data formatting is done in hardware in this coupler. This coupler provides the proper format for the following modes: uncoded, block coded, or convolutionally coded (rates 1/2, 1/3, or 1/4, frame synchronized or unsynchronized).

3. Decoded data buffer (DDB). This functional subassembly provides the hardware data formatting and the selector channel interface required for efficient transfer of data from the DDA CPU to the Telemetry and Command Processor (TCP). A single 16-bit control word allows the program to define DDA CPU core memory areas as sets of characters of 1, 5, 6, 8, or 16 bits, as well as define the number of characters to be packed in a 24-bit word and the number of trailing zeroes to be appended to each 24-bit word.

4. TCP/DDA coupler. This functional subassembly handles all communications between the TCP and DDA. It contains the circuitry required to decode the I/O controls generated by the XDS-920 in the TCP and perform the required input/output operations. Transfers of data from the TCP to the DDA are accomplished through the use of interrupts to the DDA CPU. The TCP can, at any time, issue a command, energize output M (EOM) instruction followed immediately by a parallel output transfer (POT instruction). The data is stored in a register in the coupler and the coupler generates one of three interrupts to the DDA CPU depending on which EOM had been issued. The interrupt processor in the DDA CPU then reads the data out of the register into the DDA CPU core memory.

The TCP/DDA coupler contains the hardware necessary to generate interrupts to the TCP. There are two interrupts which the DDA can generate. The generation of these interrupts is controlled via the interrupt status word (ISW), which is a 16-bit hardware register in the TCP/DDA coupler. The ISW is loadable by the DDA and readable by the TCP (via a dedicated command and parallel input sequence). Interrupts to the TCP are generally generated whenever the ISW is loaded by the DDA. The two most significant bits of the ISW are used to determine which of the two interrupts is to be generated. The remainder of the bits can be used as a message to the TCP concerning the interpretation of the newly generated interrupt. In this manner, it is possible for the two available interrupts to be used for multiple functions.

Data transfers from the DDA to the TCP are always via the DDB. The normal procedure is for the DDA CPU to issue a format command to the DDB, set up

the DDB selector channel and then load the ISW, thereby generating the proper interrupt to the TCP. The TCP response is to take the data as fast as it can via its parallel input channel. The TCP coupler monitors the TCP parallel input activity and controls a TCP busy flag which stops the selector channel whenever the DDB data registers are full and there is a word ready for transfer to the TCP.

5. Interrupt coupler. This subassembly provides the voltage level conversion circuitry for the 24-line parallel input bus and interrupts to the TCP and the interrupts associated with devices connected to the TCP emulator. This module also provides the acknowledge interrupt circuitry for interrupts generated by the FTS/DDA and TCP/DDA couplers to the DDA CPU. The DDA master clock is also located on this module.

6. TCP emulator (at 64-m antenna sites only). This functional subassembly is built on two circuit panels and provides four identical I/O channels which are electrically indistinguishable from the parallel input/parallel output (PIN/POT) channels of the XDS-920 computer used in the TCP. Special firmware is provided to operate the TCP emulator, providing emulation of the four XDS-920 parallel I/O instructions. The TCP emulator makes it possible to plug a Block Decoder Assembly (BDA), HSDL/WBDL, or Symbol Synchronizer Assembly (SSA) into the DDA without hardware modification.

III. DDA Firmware

The primary factor involved in the choice of the Interdata computer was the computational speed and flexibility available through microprogramming. A set of fifteen user-defined instructions has been built into the processor's read-only memory which allows the processor to perform relatively complex computations such as the sequential decode and tail correlation for frame synchronization acquisition at a rate which is approximately six times faster than the equivalent computation could be done using the standard instruction set. In addition to the extra instructions, the processor's interrupt-handling firmware has been modified to include the option of treating external interrupts on a priority basis. The special priority interrupt system firmware provides the queuing and servicing of interrupt processes in order of priority as defined by the interrupting device address. In this section this special firmware is briefly described.

A. TCP Autoload

This instruction is used to transfer an entire program from the TCP to the DDA. It is intended to be executed in the event of the occurrence of a program load interrupt from the TCP. Once execution of this instruction has started, the processor is put into a loop testing for the occurrence of program load interrupts. Each program load interrupt causes the processor to input a halfword (16 bits) from the TCP/DDA coupler. The first two halfwords are treated as beginning and ending addresses. Subsequent halfwords are stored in consecutive memory locations starting at the beginning address. The instruction terminates when a halfword is stored in the ending location. All other external interrupts which occur during the execution of this instruction are acknowledged but no action is taken.

B. Compute Tail Correlation

This instruction makes use of the so called "quick look" property of the class of convolutional codes currently being used to compute the likelihood that a given position in the received symbol stream is the end of a frame of coded data. Repeated execution of this instruction at all possible positions in the symbol stream is sufficient to find frame synchronization with arbitrarily high confidence.

C. Sequential Decode

This instruction implements the sequential decoding algorithm. It is necessary that there exists a properly formatted data buffer containing the received symbols and tail sequence associated with one spacecraft data frame. It is also necessary that the processor's general registers be loaded with all the parameters required by the instruction such as the location of metric tables, the impulse response, and the tail length. The execution time of this instruction is a variable depending upon the frame size and the details of the noisy received data. It is therefore necessary that this instruction be interruptable. This is accomplished by having the sequential decode firmware periodically test for the presence of an external interrupt. If an interrupt is detected, the firmware stores some of the processor microregisters in memory and does a premature exit with the location counter pointing to the *interrupt return* instruction. After the interrupting process has been completed, the *interrupt return* instruction is executed. *Interrupt return* is another user-defined instruction which restores the microregisters and transfers control back to the *sequential decode* instruction for continuation. Upon normal

completion of the *sequential decode* instruction, the location counter is incremented sufficiently to skip over the *interrupt return* instruction and the associated microregister storage area.

D. Conditionally Or Block

This instruction is used to load tail sequence into the received data buffer prior to the execution of *sequential decode*. It can also be used to add the comma-free vector into received data buffers for the block decode. The instruction performs a logical "exclusive or" of a programmable data mask into a buffer of up to 32 consecutive memory locations conditioned upon presence of ones in a 32-bit programmable register.

E. TCP Emulator Instruction

There is a set of four user-defined instructions which operate in conjunction with the TCP emulator hardware to completely emulate the PIN–POT I/O channel of the XDS-920 computer used in the TCP.

F. Halfword I/O Instructions

There is a set of four I/O instructions which are used to initiate data transfers over the 16-bit I/O channel between external devices and memory. These instructions are the counterparts to the standard I/O instructions which are 8-bit byte oriented.

IV. DDA Operational Software

Although the DDA has considerable special-purpose hardware and firmware to assist in performing the required functions, the primary control of the DDA is implemented in software. In this section, the operational program which implements the sequential decode function is discussed in order to illustrate the role that is played by software in the DDA. Figure 2 is the functional block diagram of the operational program which is used in the sequential decode mode. The various blocks (except the DDA Executive) shown in this figure may be thought of as subprograms. The system is implemented by having the DDA Executive Loop continuously checking for enabled subprograms and executing them when found. In general, subprograms may be enabled by interrupt processors or by the execution of other subprograms. The following is a brief description of each of the blocks.

A. Memory Fill

This subprogram is enabled by the occurrence of a *program load interrupt* and *initialization interrupt*. (These interrupts are generated by the TCP coupler, see *Section II-B-4*.)

B. Acquire Frame Synchronization

This subprogram is enabled under any of the following conditions: The whole system is initialized (such as at the beginning of a pass), the sequential decoder has declared itself out of synchronization due to excessive erasures, or the TCP has commanded that the sequential decoder go out of synchronization due to an anticipated data rate change. Once enabled, this subprogram acquires frame synchronization by repeated execution of the *compute tail correlation* instruction (see *Section III-B*) at all possible positions in the input system stream. The likelihood functions obtained in this fashion are retained and compared against a series of threshold values in order to find a likelihood value that implies a probability of synchronization which is sufficiently high to declare that synchronization has indeed been acquired.

C. Sequential Decode

This subprogram decodes a frame of convolutionally encoded data. It is enabled whenever frame synchronization has been acquired and a new frame of data properly formatted for sequential decoding has been loaded into core by the SSA coupler. This subprogram sets up the general registers and executes the *sequential decode* instruction. Normally, the *sequential decode* instruction is completable and upon completion this subprogram sends a message to the I-1 queue that a frame of decoded data is ready for transfer to the TCP. In the event that the *sequential decode* instruction is not completable (due to an excessively noisy frame of data), the *sequential decode* is artificially terminated and the *erasure* subroutine executed.

D. Erasure

This subroutine is executed when the *sequential decode* subprogram is unable to decode a frame in the allotted time. This subroutine sends a message to the I-2 queue that there is an erased frame ready for transfer to the TCP. If this subroutine is executed three consecutive times without a successful *sequential decode*, the decoder is declared out of synchronization and the *acquire synchronization* subprogram is enabled.

E. Memory Recall

This subprogram is enabled by the *memory recall interrupts* from the TCP. Recalled areas of DDA core are prepared for transmission over the I-2 interrupt lines.

F. I-1 Queue and I-2 Queue

These two subroutines control the transfer of data to the TCP by stacking I-1 and I-2 transfer requests. I-1 is reserved for decoded data; I-2 is used for all other transfers such as erasure data and special data requested by the TCP.

G. DDA Executive Loop

The flow chart of the DDA Executive Loop is shown in Fig. 3. Since all data transfers to the TCP must occur through the DDB, I-1 and I-2 transfers are mutually exclusive—priority being given to decoded data over the I-1 interrupt line. Each cycle of the Executive Loop begins with monitoring of the transfer queues and initiation of transfers when possible. Subprograms are then accessed on a priority basis and executed when enabled. When all enabled subprograms have been executed, the Executive Loop begins a new cycle.

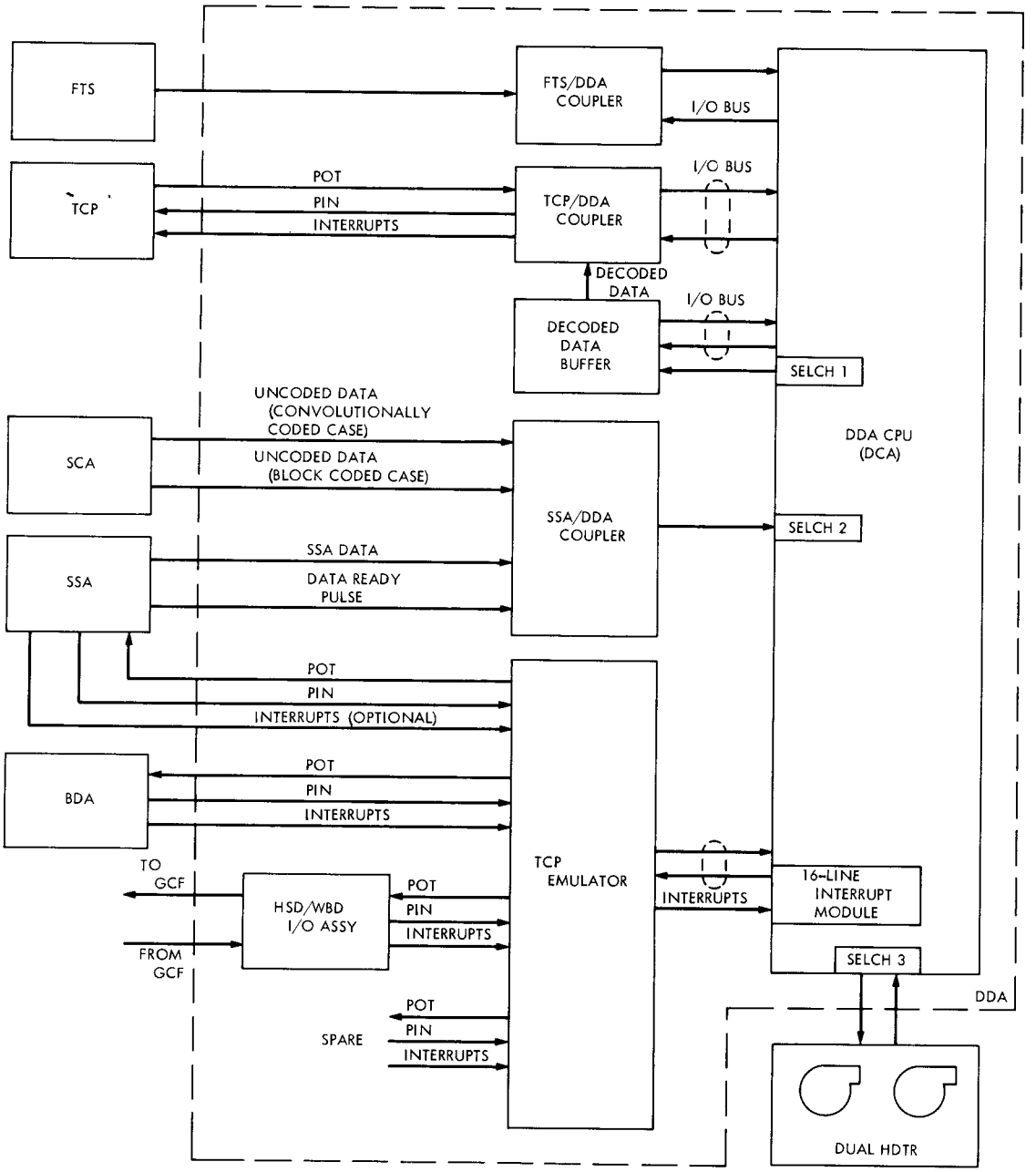

Fig. 1. Data Decoder Assembly block diagram

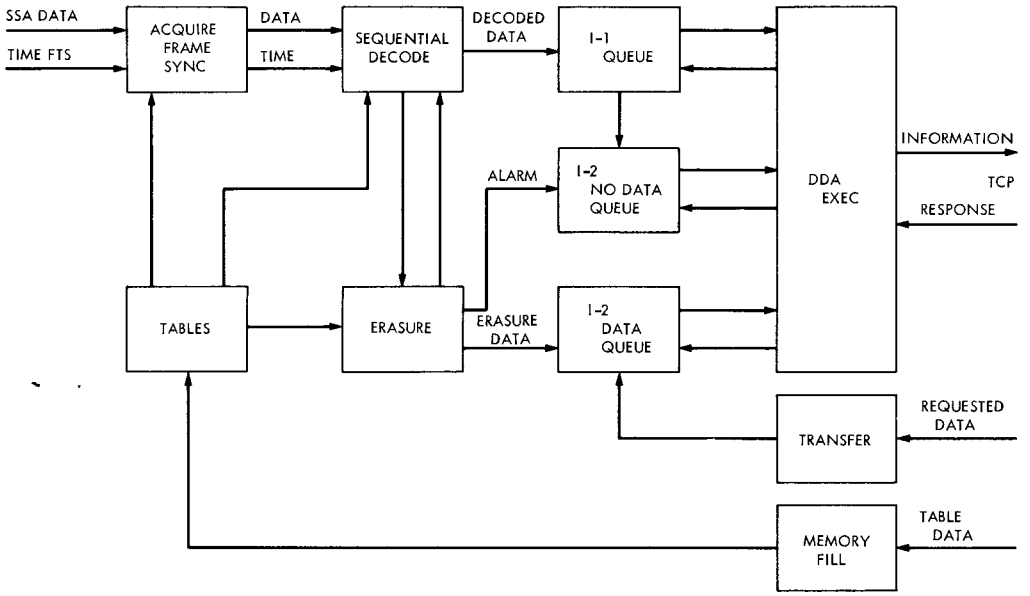

Fig. 2. DDA operational program functional block diagram

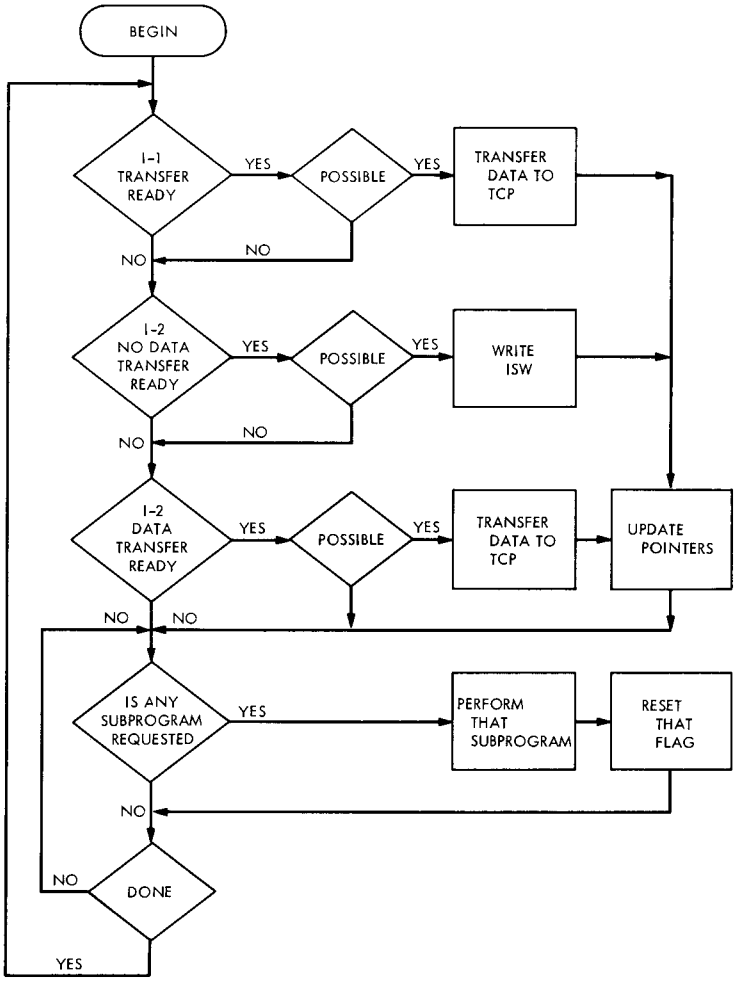

Fig. 3. DDA Executive Loop flow chart