SOFTWARE SUPPORT FOR A STORED PROGRAM PCM DECOMMUTATOR

Dwight M. Peterson
Fleet Support Department
Fleet Analysis Center
Corona, CA 91720

ABSTRACT

Telemetered data generated by missile systems has become increasingly complex with the inclusion of asynchronous data streams, variable word lengths, and discrete encoding. The display of this data for analysis purposes requires sophisticated equipment, usually designed with a programmable architecture. This paper describes software support that was developed for a stored program PCM decommutator. The software includes a cross assembler and supports downline loading of the decommutator from a host computer.

INTRODUCTION

A stored program PCM decommutator accepts input telemetry data and obtains frame subframe, and sub-subframe synchronization by utilizing internally stored instructions. The decommutator assigns Identification Tags (ID TAGs) to any or all syllables in each frame, optionally performs arithmetic logic operations, and selectively outputs data to a remote computer or Digital to Analog Converters (DACs). To use the decommutator to process data generated by a specific telemeter, machine language instructions must be generated.

Programming for the decommutator is done at the machine language level in hexadecimal format. As decommutator programs increase in length and become more complicated, a mnemonic language becomes increasingly desirable to assist a programmer in writing programs.[(1)]

The Fleet Analysis Center has developed a cross assembler for the Aydin Monitor Systems Model 1126B stored program PCM decommutator, together with a program which downline loads machine language instructions to the decommutator from a host computer.

THE CROSS ASSEMBLER

An assembler allows a mnemonic based language to be used when programming at the machine level. The cross assembler developed by the Fleet Analysis Center was written in FORTRAN IV to run on a PDP-11 minicomputer. It will automatically assign memory locations, compute branch offsets, assure that data fields are within a specified range, and generate machine language instructions for the decommutator.

The cross assembler makes two passes through the source decommutator program. The main purpose of pass 1 is to build a local symbol table and perform a rudimentary syntax check of each source statement. During pass 2, the cross assembler simultaneously writes the machine language code it generated to an output file and creates an assembly listing. Errors found during the assembly process are flagged with an error description that is included in the listing.

The decommutator source program is composed of a sequence of source coding lines. Each line contains a single assembly language statement. The assembly language statement may consist of as many as four fields. These fields are identified by their order of appearance within the assembly language statement. The format of the assembly language statement is:

Label: Op-code Operands ; comments

The label and comment fields are optional. The op-code and operand fields are interdependent; when both fields are present in a source statement, each field is evaluated by the cross assembler in the context of the other. Each of the four fields, if present, uses the tab character as a delimiter.

A label is a means of symbolically referring to physical memory location within a program. When a label is used in a decommutator program, both the label and the value of the current location counter are entered into the symbol table. The current location counter is a means by which the cross assembler assigns memory addresses to the source program statements as they are encountered during the assembly process. Memory addresses assigned by the cross assembler are absolute and represent the actual physical address used by the decommutator.

Two operating modes exist within the cross assembler. The first is used to assemble decommutation instructions (Decom mode), the second to assemble computation module instructions (Comp mode). The cross assembler defaults initially to Decom mode, and the ENA directive is used to change modes to Comp mode. Two modes are required because

address calculation is performed differently for decommutation and computation module instructions.

Programs for the decommutator are written in assembly language for all missile telemetry formats. As display requirements for a particular telemetry format change, the source code is modified and processed by the cross assembler to generate a new machine language program.

The instructions used by the cross assembler are listed in Table I.

TABLE I. Instructions Used By The Cross Assembler

I. DECOM INSTRUCTIONS

OP Code	Operands	Description
DST	C, P, L/M, LENGTH, IDTAG	Distribution
JMP	DISPLACEMENT	Jump
SF1	I, S, DISPLACEMENT	Jump to subframe #1
SF2	I, S, DISPLACEMENT	Jump to subframe #2
SF3	I, S, DISPLACEMENT	Jump to subframe #3
SF4	I, S, DISPLACEMENT	Jump to subframe #4
SFS	I, S, DISPLACEMENT	Jump to subframe #5
RET		Jump return

II. COMP INSTRUCTIONS

OP Code	Operands	Description
SCL	C, CND, +/-, SCALE, OFFSET	Y=M(X±B) scaling
OFS	C, CND, +/-, OFFSET	Y=X±B offset
SUM	C, CND	Running Summation
OB1	C, LOC, MASK	Code conversion
OB2	C, LOC, MASK	Code conversion
OBS	C, LOC, MASK	Code conversion
AND	C, VALVE	Logical AND
OR1	C, MASK	Logical OR
XOR	C, MASK	Logical exclusive OR
NAN	C, MASK	Logical NAND
NOR	C, MASK	Logical NOR
NXO	C, MASK	Logical Not EX OR

MAC	C, VALVE, MASK	Mask and compare
COM	C, VALVE	Compare
DLC	C, CND, F/D, EOF, FMT/LOC	Decom list change
DEP	C, ASRC, AREG, ADST, BSRC, BREG, BDST	Discrete event
DBO	C, CND, DTRANS, REG, DREG, BSRC, BDST	Discrete proc
BRN	C, CND, ADDRESS	Conditional branch
NOP		No output

III. ASSEMBLER DIRECTIVES

OP Code	Operands	Description
ENA	DECOM/COMP	Enable Decom/Comp mode
LST	ADDRESS	Change location counter address

MODEL 1126B DECOMMUTATOR ARCHITECTURE

The Model 1126B PCM decommutator[(2)] is controlled by machine language instructions stored in an internal Random Access Memory (RAM). Two groups of instructions exist: Decommutation Instructions and Computation Instructions. The first group of instructions is used for the decommutation of incoming data and the second is used to manipulate the data in the computation module.

Decommutation Instructions are stored in an 8K x 24 bit RAM memory. This will be referred to as the Decom memory. Instructions in the Decom memory are grouped in a sequential list, called a main frame distribution list, in which each instruction defines a word in the PCM telemetry format. The first instruction in the sequential list corresponds to the first word in the telemetry format, the second instruction to the second word, and so on. The list continues until the last word in the format is defined; by convention, the last word is the Frame Sync Word. A word in the telemetry format which is not subcommutated is programmed with a distribution instruction. The distribution instruction defines the Least Significant Bit (LSB)/Most Significant Bit (MSB) alignment and the number of bits in the word, assigns a unique Identification Tag (ID TAG) to the word, and determines whether the word should be ignored, passed to the computer output port, or passed to the computation module. A word in the telemetry format which is subcommutated is programed with a Jump to Subframe instruction. The Jump to Subframe instruction gives the address of a sequential list in the Decom memory which contains additional instructions required to decommutate the subframe. The sequential list, called a subframe distribution list, usually contains distribution instructions. Supercommutation is handled by using multiple distribution instructions which assign the same ID TAG to each supercommutated word.

Computation module instructions are stored in an 4K x 24 bit RAM memory. This will be referred to as the Comp memory. Computation module instructions are used to manipulate the incomming data by shifting, logic operations, code conversion, etc.

PROGRAMMING EXAMPLE

Figure 1 shows, in diagram form, the SM-2 FTR missile telemeter format. Each block represents a word in the telemetry format which is identified by a function number located inside. The function number is assigned arbitrarily to represent telemetered missile functions like vertical acceleration, tail position, etc. The widths of the blocks in Figure 1 indicate the lengths of the words; a narrow block indicates an 8-bit word, and a wide block indicates a 12-bit word.

The beginning of the main frame distribution list required to decommutate the FTR missile telemetry format is shown in Table II for the first 24 words utilizing cross assembler instructions.

The first word in the FTR telemetry format is an Identification (ID) counter. The counter increments from 0 to 5 by 1 every 108 words and is used as an index to define the location of subcommutated words. The first word is programmed with a SFI (jump to subframe #1) instruction because it contains subcommutated data. The SF1 instruction defines word 1 as having the ID synchronization counter (S present) and defines the mnemonic S001 which represents the address of the subframe distribution list.

The second word is programmed with a DST (distribution) instruction which defines the word as being LSB first, 8 bits in length, and assigns the value associated to the mnemonic F01 as the ID tag. The cross assembler will scan its symbol table and insert the numeric value associated to the mnemonic F01.

Words 3 through 13 are programmed in a manner identical to that used in programming word 2 except that different mnemonics are used to define the ID TAG field of the distribution instruction.

The distribution instruction used to decommutate word 14 is identical to that used for word 2. This is beacuse function number 1 is supercommutated and appears in words 2, 14, 26,...,etc. Supercommutated data is processed by the decommutator by haveing the same ID TAG defined for all words which contain identical information.

TABLE II. Beginning Of Main Frame Distribution List

Mem Adr	DECOM Instr	Label	OP Code	Operands	Comments
0200	9 8 0062		SF1	,S,S001	;Word 1 ID Count Subframe
0201	5 7 018C		DST	C,,L,8,F01	;Word 2 Function 1
0202	5 7 018F		DST	C,,L,8,F02	;Word 3 Function 2
0203	5 7 0192		DST	C,,L,8,B	;Word 4 Function B
0204	5 7 0193		DST	C,,L,8,F03	;Word 5 Function 3
0205	5 7 0194		DST	C,,L,8,F04	;Word 6 Function 4
0206	5 7 0195		DST	C,,L,8,F05	;Word 7 Function 5
0207	5 7 0196		DST	C,,L,8,F06	;Word 8 Function 6
0208	5 7 0197		DST	C,,L,8,F07	;Word 9 Function 7
0209	5 7 0198		DST	C,,L,8,F08	;Word 10 Function 8
020A	5 7 0199		DST	C,,L,8,F09	;Word 11 Function 9
020B	5 7 019A		DST	C,,L,8,F10	;Word 12 Function 10
020C	5 7 019B		DST	C,,L,8,F11	;Word 13 Function 11
020D	5 7 018C		DST	C,,L,8,F01	;Word 14 Function 1
020E	5 7 018F		DST	C,,L,8,F02	;Word 15 Function 2
020F	9 4 0063		SF1	I,,S016	;Word16 Subcom WD16
0210	9 4 0069		SF1	I,,S018	;Word18 Subcom WD18
0211	9 4 006F		SF1	I,,S019	;Word19 Subcom WD19
0212	9 4 0075		SF1	I,,S021	;Word21 Subcom WD21
0213	9 4 007B		SF1	I,,S022	;Word22 Subcom WD22
0214	9 4 0081		SF1	I,,S024	;Word24 Subcom WD24

Word 16 is programmed with a SF1 instruction much like word 1. However, word 16 contains information which must be indexed by word 1 before it has meaning. When the ID counter in word 1 has a value of zero, then word 16 contains function G1. If word 1 contains a one, then word 16 contains function G23. The SF1 instruction that describes word 16 defines an Indexed branch (I present) to the subframe distribution list defined by the mnemonic S016. The 1126B decommutator will take the address defined by the mnemonic S16 and add to it the value of the current ID counter defined by word 1, and fetch the correct distribution instruction from the subframe distribution list. The subframe distribution list for word 16 is programmed as shown in Table III.

TABLE III. Subframe Distribution List For Word 16

Mem Adr	DECOM Instr	Label	OP Code	Operands	Comments
0263	5 B 019C	S016:	DST	C,,L,12,G001	;G001 ID count 0
0264	5 B 019D		DST	C,,L,12,G023	;G023 ID count 1
0265	5 B 019E		DST	C,,L,12,G045	;G045 ID count 2
0266	5 B 019F		DST	C,,L,12,G067	;G067 ID count 3
0267	5 B 01A0		DST	C,,L,12,G089	;G089 ID count 4
0268	5 B 01A1		DST	C,,L,12,G111	;G111 ID count 5

After all main frame and subframe distribution instructions required to describe a telemetry format have been defined, the cross assembler is placed in comp mode by the ENA directive. The mnemonics that defined the ID TAG field of the distribution instructions used in the main frame and subframe distribution lists above appear as labels for computation module instructions. This is illustrated by the cross assembler instructions in Table IV.

The LST directive is used to change the cross assembler location counter. The functions F01 and F02 are converted from 2's complement to offset binary and output to Digital to Analog Converters (DACs) 1 and 2, respectively. The remaining functions are defined but use a No Output (NOP) instruction to inhibit DAC output.

Defining the mnemonics for all distribution instructions in the computation module section of the source code allows the cross assembler to automatically generate ID TAG values. All mnemonics are defined, whether or not the function will be output to DAC's or requires the processing capabilities of the computation module.

During the assembly of the source code, the cross assembler saves all mnemonics and the numeric values assigned to them in a symbol table. Table V is the symbol table for all the above decommutator instructions.

After a decommutator program has been assembled, a listing of the source code and a downline load file are generated. The format of the listing is identical to that of the programming examples shown above. The downline load file contains the physical memory address and the decommutator machine language instruction in hexadecimal. The downline load file for the main frame distribution list programming example would appear as shown in Table VI.

TABLE IV. Computation Mode Cross Assembler Instructions

Mein Adr	DECOM Instr	Label	OP Code	Operands	Comments
2000			ENA	COMP	;Enable Comp Mode
218C			LST	0218CH	;Start at 0218CH
218C	7 A 0080	F01:	XOR	C,080H	;Sign
2180	3 9 8C01		SHF	C,,L,8,D,A,1	;Output DAC 1
218E	B 0 0700		NOP		;Return
218F	7 A 0080	F02:	XOR	C,080H	;Sign
2190	3 9 8C02		SHF	C,,L,8,D,A,2	;Output DAC 2
2191	B 0 0700		NOP		;Return
2192	B 0 0700	B:	NOP		;Return
2193	B 0 0700	F03:	NOP		;Return
2194	B 0 0700	F04:	NOP		;Return
2195	B 0 0700	F05:	NOP		;Return
2196	B 0 0700	F06:	NOP		;Return
2197	B 0 0700	F07:	NOP		;Return
2198	B 0 0700	F08:	NOP		;Return
2199	B 0 0700	F09:	NOP		;Return
219A	B 0 0700	F10:	NOP		;Return
219B	B 0 0700	F11:	NOP		;Return
219C	B 0 0700	G001:	NOP		;Return
2190	B 0 0700	G023:	NOP		;Return
219E	B 0 0700	G045:	NOP		;Return
219F	B 0 0700	G067:	NOP		;Return
21A0	B 0 0700	G089:	NOP		;Return
21A1	B 0 0700	G111:	NOP		;Return

A program was written which reads the contents of the downline load file and transfers the instructions to the decommutator via a contractor supplied computer interface. The program performs a load followed by a verify for all instructions contained in the downline load file. Verify errors cause the memory address, expected and found value to be output on the user terminal.

TABLE V. Symbol Table

Mnemonic	Decimal Value	Hexadecimal Value	Mnemonic	Decimal Value	Hexadecimal Value	Mnemonic	Decimal Value	Hexadecimal Value
B	00402	0192	F06	00406	0196	G001	00412	019C
F01	00396	018C	F07	00407	0197	G023	00413	019D
F02	00399	018F	F08	00408	0198	G045	00414	019E
F03	00403	0193	F09	00409	0199	G067	00415	019F
F04	00404	0194	F10	00410	019A	G089		01A0
F05	00405	0195	F11	00411	019B	G111	00417	01A1
						S016	00099	0063

TABLE VI. Downline Load File For Main Frame Distribution

Mem Adr	DECOM Instr	Mem Adr	DECOM Instr
0200	9 8 0062	020B	5 7 019A
0201	5 7 018C	020C	5 7 019B
0202	5 7 018F	020D	5 7 018C
0203	5 7 0192	020E	5 7 018F
0204	5 7 0193	020F	9 4 0063
0205	5 7 0194	0210	9 4 0069
0206	5 7 0195	0211	9 4 006F
0207	5 7 0196	0212	9 4 0075
0208	5 7 0197	0213	9 4 007B
0209	5 7 0198	0214	9 4 0081
020A	5 7 0199		

REFERENCES

1. Wester, John G., and Simpson, William D., 1976, Software Design for Microprocessors, Texas Instruments Incorporated.

2. Aydin Monitor Systems, March 27, 1978, Operation and Maintenance Manual for Model 1126B PCM Decommutator.

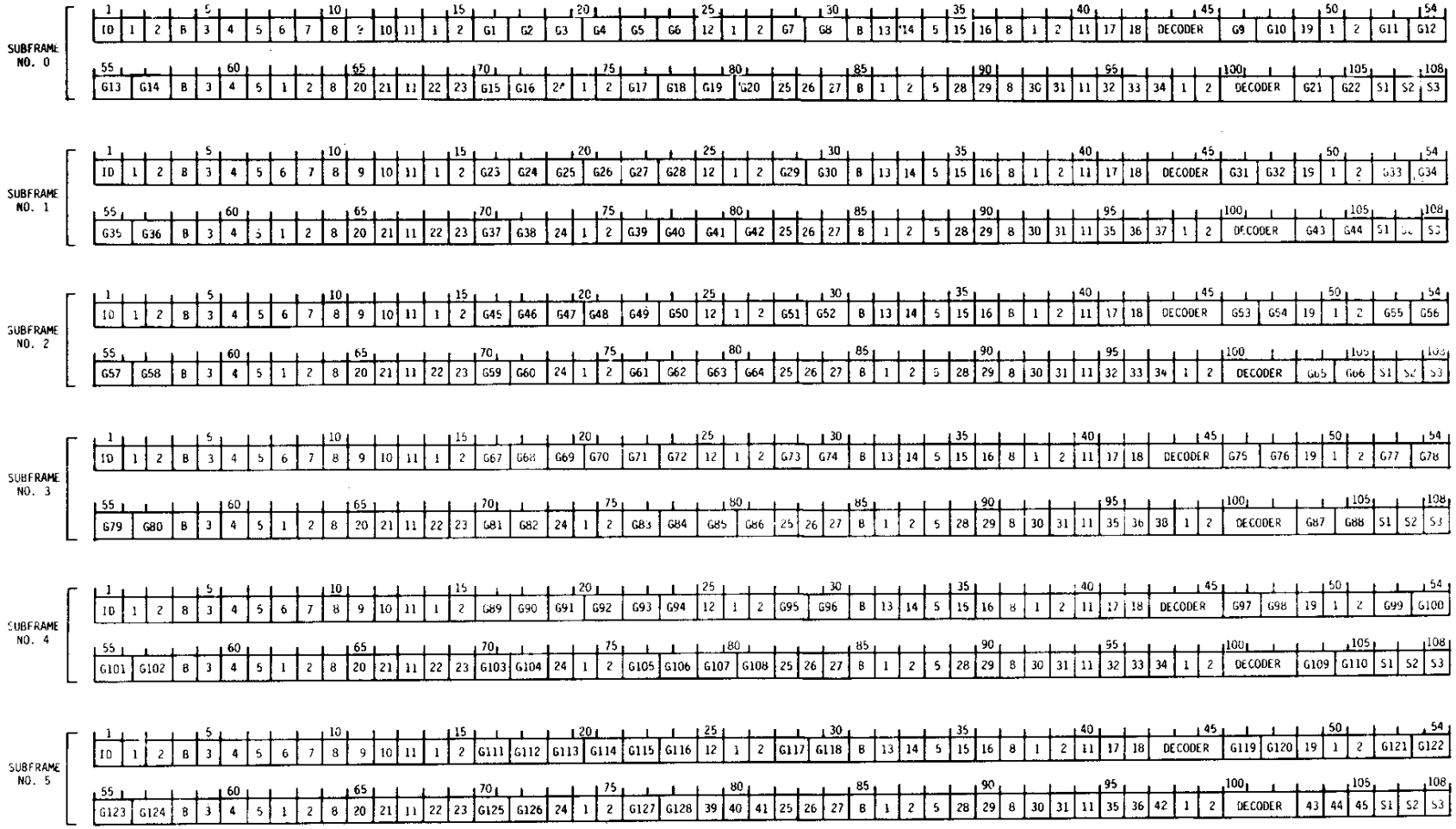

FIGURE 1. SM-2 FTR MISSILE TELEMETRY FORMAT