C H A P T E R IV

DESCRIPTION OF SYSTEM IN LRN MODE.

The present discussion elaborates the environment supported by the system for user to create software structures of desired control instrumentation. Programming level offered by the system has also been described.

As an overview of the system-software support, the system allows user to create programs under as many as 10 indices. All these areas could be opened irrespective of order of the index and the flexibility offered allows user to develope a program segment within an area and complete it, after developing a segment in any other area.

The programming levels offered is very much similar to the Microsoft Inc. Basic - 86 interpretable version(38), however a few curtailments and/or additions have been incorporated keeping in mind the control environment of the instrumentation (sec. 2.1). Further, the facilities offered by the system are much more exclusive compared to the version of Basic implemented on kits, e.g. MPF - I Basic(39) (CPU Z-80, 4 KB version) and Tiny Basic(40) (CPU 8085, 10 KB version). The language allows arithmatic manipulation on signed integer or signed real numbers, and a data area allows maximum 6656 variable reference to be used. Four such independently accessible data areas are supported.

4.1. Input And Output Statements.

Multifold accesses offered by the system allow user to access numeric entries from keyboard (KB) to be referenced there further with a variable name or a digital entry from KB (key codes) and to be loaded in register AL of the CPU. Additionally 8 bit or 16 bit ports could be directly accessed in accumulator or with a variable reference. Further analog data from specified channel also could be accessed in register AL or under variable reference directly. At present this facility is limited to 8 bit ADC and for 8 channels.

Output facility allows transmitting 8 bit or 16 bit data from accumulator or memory to any desired port. Further desired messages and/or numeric values also could be transmitted to the display. The numeric quantities transferred to the display are back converted from their internal forms of representation. Real numbers are displayed as floating point numbers while the integers are outputted as hex nibbles with their sign. Messages are constraint to follow the numeric values, if both combined together form field of the current display.

Syntax associated action group, subgroup and corresponding actions have been tabulated in Table 4.1. A point of observation could be the comparison of Table 3.4 and 4.1. The action groups associated with these statements, except for in case of END statement, is a left shifted byte that corresponds to the function key entries. While the subgroups are allocated in view of ease indifferentiation of action sequences at run time.

4.2. Control Statements.

Conditional statements form a group of measure emphasis. All non-Boolean conditional operators have been accomodated in the present structure and are listed in Table 4.2. Not only comparisons between two variables or a variable and immediate numeric entry is allowed but comparison between bytes accessed from the addressed port with a variable (memory) or a numeric constant is also allowed. Similarly bytes accessed from ADC ports could also be compared. To manipulate program flow, comparison of a entry from KB is given a separate status, which relieves user from entering actual port address of 8279. In this statement comparison with immediate bytes only is allowed. Syntax, associated group, subgroup and action are tabulated in Table 4.3. The policy adopted for associating groups and subgroups is the same as discussed in sec.4.1.

The software supports use of FOR - NXT type of structure where the parameter of eteration is allowed to be any 8 bit integer variable, without index. (Scheme of using variable names and data memory management are discussed further in this chapter). The system allows a stack area of maximum 512 bytes (Table 3.1), which results to be a constraint on the nesting of FOR - NXT structures. Though actual order of nesting depends upon the length of arithmatic statement in the structure. As no brackets are allowed in the defination of arithmatic expressions, the 6 to 7 order of nesting would not be a over estimate. Remaining conditions of nesting are similar to standard Basic version$^{(38)}$. The allowed range of parameter variation in this structure is between 0 to +127. The syntax and associated group is tabulated (Table 4.3).

Unconditional branching is allowed through GTO (GO TO), statement. At the time of interpreter code conversion the jumps are estimated whether back or forth and from subgroups of GTO statement. The syntax, associated group, subgroup and actions for GTO statement also listed in Table 4.3.

GSB (GO SUB), RET structure is also supported for coding subroutines and managing calls to these modules. At present subroutines from the same program area could be called and not from any other (Refer sec. 5.).

As per strategy adopted to store the interpreter code (IC) the IC blocks of all the programs are stored in the RAM without attempting to arrange the program in ascending order at the LRN time. Further the program sequence is arranged temporarily in the ascending order at the run times (sec. 5.). Therefore, accommodating inter program area subroutines is postponed for later stages of development of the system.

4.3. Allowed Data Types And Variable Allocations.

Within Intel 8086/8088 systems trade-offs of hardware overhead and time and code implementation lead to the adoption/exclusion of Maths processor[41] Intel 8087, within the domain of specific requirements. For the sake of comparison the arithmatic function which eat up 2 ms of time on 8086 are executed in a few microseconds by 8087. Further arithmatic logic units like Intel 8231, 8231A, utilising Chebyshev polynomials[42,43] for implementing the ALU algorithms, or 8087 do offer a choice of representing number in the formats offered by 8087. Further if 8087 is excluded from the system Intel library listed opcode modules are linked to higher level language programs for numeric crunching[5].

As the present system is intended to be added - on to a PC, while defining/implementing the data formats a care is taken not to reduce the system through-put drastically. This has been achieved as at the cost of allowed ranges numbers and limited choice of accuracy. If additional accuracy is intended the data items could be very well be down loaded to the PC and operated standard formats through higher level languages.

Allowed data formats, range, accuracy and length are tabulated (Table 4.4). With a sufficient care and valid approximations control are line fitting routines could be developed with these data formats. As a sample case a 12 bit intelligent self tuning, recurssive PID algorithm acting on a single variable is in the process of developed. Syntax of the number entries within these categories is also given in Table 4.4.

Regarding use of variable names, characters from A to P represent real numbers while characters from Q to Z represent integers; no over ride is permitted. The variables could be indexed and the dimensions allowed are 0, 6, 16 or 256. The variables without index are referred to as Individual elements and could be only of 16 integer or 24 real types. The variable with dimension 6 could be 8 integer or 16 real type. 16 or 256 dimensional arrays could be of any declared types. Further except for 6 dimensional (default variables) all other variable types could be from a specified data area. Maximum 4 such data areas could be used and variable references within these are mutually independent. Further a program, with its area index, is restricted to use and specify a single data area only.

The default area is common to all programs and need not be specified while entering a program segment. After selection of a program area 5 statements specify (1) Index of data area (between 1 to 4), (2) Variables with dimension 256, referred as Extended Area, (3) Variables with dimension 16, referred as Normal Area, (4) 16 Integer variables and (5) 24 Real variables. Syntax of these statements is given in Table 4.5. Combination of statements (2) and (5) specify 16R and 24R extended area variables while combination of 2 and 4 specify 8I and 16I Extended Array variables. Further, variables within Normal Array declaration are also differentiated on the similar lines of extended arrays. This has been emphasized with an example in Table 4.6.

4.4. Arithmatic Operations.

At present only + (addition), - (subtraction), * (multiplication), / (division) operations are supported. The key statement LET definesthe group. Variable on the left hand side could be of any type defined in Table 4.4. While the variable on the right hand side have to be either real or integer and of the type allowed. The allowed types of variables on RHS for a variable type on LHS is tabulated in Table 4.7. Additionally the table specified conversion to take place before actual value for the variable on LHS is loaded in memory, these conversion operators are determined and loaded as a byte in the IC field. Subgroups are decteted by the variables on RHS. On RHS if real type of variablesare accommodable then the real are also allowed but not vice-versa. Similarly allowed 16 integer type on RHS accomodated 8 integer type of variables but not vice-versa. This leads to four subgroups of LET statement viz. 1) 24 Real or 16 real (011B) !, 2) 16 real (100B) !, 3) 16 integer or 8 integer (101B) !, 4) 8 integer (110B) ! types. Syntax of LET statement is given in Table 4.8.

Another form of LET statement is used to call user written target program and is referred as LET-CONVERT at present upto 16 target sub programs could be refered through the language. The convert routines are mainly intended for determination of parameters of physical, where transduced data is acquired through ADC ports. Therefore, two types of convert statements, one operating on implicit accumulator to provide input data and another where a variable reference provides the input data are supported (Table 4.1, ,). Matching data type on LHS and RHS is left to the target subroutine themselves. Syntax of LET-CONVERT is also given in Table 4.8. INR (increment) and DCR (decrement) operating on integer variable only are also supported and are designed to provide sequential indices while acquiring data sets of experimental parameters, e.g. Thermo emf and Tcs, differential temperature, Tc, with time (Q and C_p measurements, sec. 2.1).

4.5. Delay Statement.

Unique feature of present language is software detectable delay time. Delay times of 999.9 seconds to 0.001 second is selectable. Only four BCD entries are allowed as delay parameter and position of the decimal point specifies whether 1 ms or 10 ms or 100 ms least count (clock frequencing) is to be used for the count down. Delay suspends normal state of operation for the specified time. Syntax group and subgroups of these statements is given in Table 4.9.

4.6. Overview Of The System Operations.

Flow chart 4.1 describes overall system operations. During POST (Power on self test) the system checks for the configuration as defined in sec. 3.2 and decides the strategy of initialization of I/O devices and memory as indicated in Chapter III. Interrupt vector table is at present assumed to be a battery backed memory or a ROM and no support to write interrupt vectors is accomodated in the present software. Further, bytes which decode the key entries for displaying the key mnemonics is also assumed loaded in permanently. The ROM occupies a page of 1 K X 8 of segment 0. Further during POST a few memory locations forming a part of reference parameter table are resetted or loaded with desired values. e.g. Configuration switch specifies to monitor that the total program and data area (RAM) available on the board and monitor accordingly loads a appropriate word in memory location referred with offset identifier 'AV_MEM_BLOCKS'. Predefine (Table 3.1) offsets in the segment_0 are reserved for monitor to create various tables for its reference at LRN or RUN time. System memory is assumed batery backed as a whole. Program area is referred in terms of block of 256 bytes, while the data area are treated in terms of blocks 128 bytes. POST displays default LRN index, data area and available memory blocks of 256 bytes for programmers reference. Further monitor confirms the program storage and if found mismatching with the reference program index table it reset all the program and data area. The POST or RESET are equivalent and leaves monitor in Level_0 (L_0) (Flow chart 4.1). At this stage the system expects to enter in one of the following action sequences. The facilities for listing or deleting the source program or reseting the data area forms a group while invoking software supports for entering or running a program becomes another group. These facilities are elaborated below.

1) Deleting a program area :- Interpreter codes corresponding to a program area are stored from top to bottom in system RAM (Table 3.1). Monitor allows pointing to a program area (index) for maximum 3 number of times, after developing programs in other areas (indices) intermittently. The program area is treated in terms of blocks of 256 bytes and the interpreter codes corresponding to program area index are stored sequentially after making an entry in Program Index Table (PIT). Thus a program area gets subdivided into maximum 3 units, where each unit has its corresponding status in PIT indicating the count of reopening the area. 'DEL' 'SEL' 'Q' 'ENTER', or 'DEL' 'ENTER', refering to all the program areas, are the two allowed syntaxes. Further, making use of PIT the monitor resets the corresponding units, rearranges the interpreter code storage and PIT and program flow resumes Level_0 (Flow chart 4.1). (Listing of software for these actions is enclosed).

2) Resetting data areas :- Similar to PIT, Data Index reference table (DIT) are also generated in the LRN mode. (described later in the Chapter). Table 4.10 indicates DIT. Now as per the command, similar to DEL command a specific non-default data area or all data areas are resetted and the remaining data areas and DIT is rearranged. The monitor returns to Level_0 (Flow chart 4.1). (Listing for software for these actions is enclosed).

3) Listing Facility :-

i) Storage of key entries : Interrupt from 8279 (KBIRQ) invokes a support which loads the key entries in Key Board Buffer (KBB) and display buffer. The flow returns to monitor upon detection of 'ENTER' key. Upon 'ENTER' the system preserves key entries in a file of width 2 K in the system RAM. If this file gets completely failed then the system down loads the file with proper identifier field, to a cassette tape recorder. The communication is achieved through 8251 and FSK modem MC 14412, described briefly in sec. 3.2

ii) Retrieval of program files : LST command specifies the index of the program monitor loads the appropriate file on the RAM buffer and displays 0th sentence of the program, again through a software call to KBIRQ. The desired part of any sentences could be brought in the display field using cursor control keys and could be altered. Processor 'ENTER' key suspends the operation and monitor returns to Level_0. A module interfacing 8251 (+ FSK MC 14412) and cassette tape recorder control port 8212_2 fig. 3.2, is too standard routine and is at present left undeveloped. Software for mainly part of action sequences is developed. (For details refer listing enclosed with the dissertation).

4) Selecting a program area :- Upon RESET or POST monitor offers a program area index as default one. If user does not intend to use the default area to develop his/her program then he is allowed to invoke any other area for program development. Monitor receives the index entered by user and scans PIT to

calculate starting segment (DS) and offset (SI) for the interpreter code storage. If the program area is opened previously then status of the current open up is recorded. If the program area is getting opened for first time then only the system allows to specify data area to work with. Further if new data area is to be specified the flow continues to Level_2 otherwise the monitor expects a program statement entry.

5) Specifying contents of an data area :- If monitor is at Level_2 (Flow chart 4.1) then it expects a index entry to specify the data area. Further, it loads 4 arrays with the entries corresponding to the data specification statements given in sec. 4.4. Combining these arrays DIT is prepared which specifies type of individual variables names used and total number of 128 byte blocks required for storage. Refering to the DIT's of remaining indices the required data area is provided by making block movement of data within memory. (Block Alter Module in listing). Use of 4 data areas allows user to develop programs using common or isolated data items depending on requirements. In essence the desired data files could be coupled to any programs.

As a separate Chapter is devoted for the discussion of facilities for running interpreter codes corresponding to a program index, we elaborate generation of interpreter codes from the key entries, in this Chapter.

6) Resolving key entries into interpreter codes :- General structure of interpreter code blocks is shown in fig. 4.11. The