TRIGONOMETRIC RESOLUTION PROGRAM FOR HYDAC*

ABSTRACT: The functional characteristics of a DOS Resolver program are described. The program utilizes the general purpose components of the Digital Operations System (DOS) of the EAI HYDAC Computers to provide flexible, easily used resolver capabilities for the trigonometric resolution of system variables generated as part of an analog computer program. By time sharing components, the program described permits up to four (4) transformations to be performed accurately and economically for either analog or hybrid simulations.

GENERAL

The need for trigonometric resolution of vector components arises when physical systems involving more than one set of axes are simulated. Important practical examples are the description of the forces acting on an airframe or space vehicle. These forces (drag, thrust, lift, gravity, etc.) are described by equations of motion most often written in a coordinate system fixed with respect to the body, which rotate with respect to earth or some space coordinate. This requires forces to be expressed in terms of components (resolved) along the selected movable axes. It may also be necessary to translate the motion of the vehicle into "earth" axes; thus a coordinate rotation is required. The requirement for trigonometric resolution may also arise from a variety of other systems, mechanical or electrical, for which small angle assumptions are not valid. Such needs are so frequent in analog computation that resolvers of some type are included in the component complement of most computer systems.

The HYDAC program to be described is an economically attractive alternative to satisfying the resolution requirements of analog computer facilities. In addition to the economics of time sharing, which the program affords, there is the added advantage of the general purpose capabilities of the digital components used to perform the trigonometric resolutions. These digital building blocks can be conveniently interconnected to perform a variety of useful functions (time delay, function generation, etc.) when coordinate resolutions or axis rotations are not required in the problem being solved.

GENERAL CAPABILITIES OF PROGRAM

The DOS Resolver program is a general purpose HYDAC program for performing trigonometric resolutions on sampled analog variables. It is implemented by the interconnection of a portion of the complement** of general purpose components in the Digital Operations System of the HYDAC Computers.

Two modes of operation are provided for the performance of any one of the following transformations on sampled 13-bit binary coded data. If the inputs are "x", "y", and "a", the following outputs are produced:

MODE 0

2-inputs (Polar to Cartesian)

$$Y = K_R(x \text{ Sin } a) \qquad r \equiv x$$
$$X = K_R(x \text{ Cos } a)$$

3-inputs (Vector Rotation)

$$Y = K_R(y \text{ Cos } a + x \text{ Sin } a)$$
$$X = K_R(-y \text{ Sin } a + x \text{ Cos } a)$$

**DOS components not used in the resolver program are available for use in performing additional computing functions.

*Trade Mark
Printed in U.S.A. 093

Mode 1

2-inputs (Cartesian to Polar)

$$R = K_R(x^2 + y^2)^{1/2}$$
$$A = K_B(\text{Arctan } \frac{y}{x})$$

K_R and K_B are scale factors.

OPERATING PRINCIPLES

Operation of the program is based on the DOS implementation of the so-called CORDIC logic*. As shown in Figure 1, if a vector R has components x and y and if ky is added to x and kx is subtracted from y then a new vector R′ has been formed which has components of

$$X' = x + ky$$
$$Y' = y - kx$$

The vector has been rotated through an angle α such that

$$\tan \alpha = \frac{[(kx)^2 + (ky)^2]^{1/2}}{[x^2 + y^2]^{1/2}} = k$$

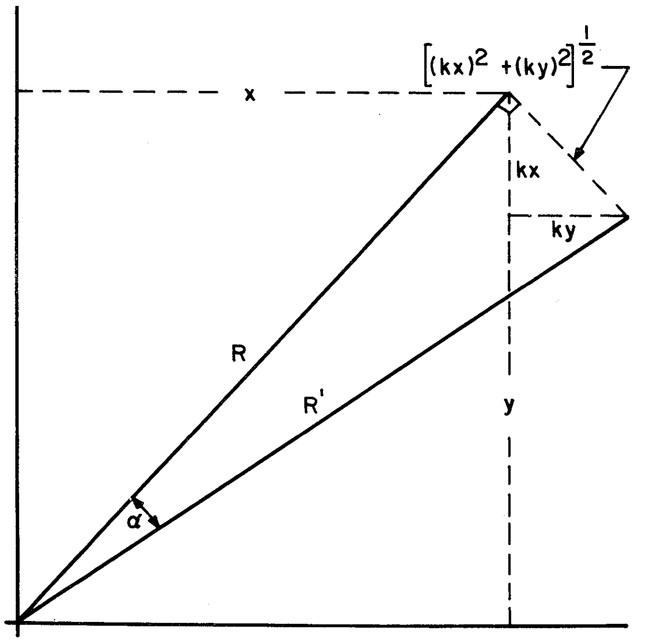

Figure 1
Diagram of Vector Rotation Scheme

*Jack E. Volder; "THE Cordic Computing Technique", Proceedings of the Western Joint Computer Conference, 1959, pp 257-261.

Note that this rotation depends only on k, and not on x and y. The magnitude of the vector is changed however, so that the new value is

$$R' = [R^2 + k^2 R^2]^{1/2} = R(1 + k^2)^{1/2}$$

which again depends only on k. The factor k can be of either sign, and so can the angle α, but the change in the vector magnitude is constant. It is convenient to take k as 2^{-i} where i is the number of increments taken, since the multiplication is easily performed in binary by right shifting.*

Cartesian–To–Polar Conversion: Implementation of the above procedures is as follows: To form a vector, the respective sampled values of its components are loaded into X and Y registers, and k selected so that y is reduced; i.e., if y is positive, k is positive; if y is negative, k is negative. In this way, y can be reduced to within 1 bit of zero in 16 steps. The number now in the X register is the magnitude of the original vector increased by the factor K_R, where

$$K_R^2 = (1 + 1)(1 + \frac{1}{4})(1 + \frac{1}{16}) \ldots (1 + \frac{1}{2^i})$$

which can be pre-determined absolutely.

The angle of rotation is obtained by adding, according to the sign of k, and by means of the A register, the fixed angles, α_i, which are stored in a serial memory unit for i steps. Thus

$$\text{Angle} = \pm \alpha_0 \pm \alpha_1 \pm \alpha_2 \pm \ldots \pm \alpha_i,$$

where Tan $\alpha_i = 2^{-i}$ and $\alpha_0 = 90°$. The first step of 90° insures that the x component of the new vector is positive, thereby facilitating the reduction of the y component to zero. It also allows a total rotation of ±180°, and is easily implemented by interchanging the contents of the X and Y registers.

Polar–To–Cartesian Conversion: To form the X and Y components from the vector r at an angle "a", r is loaded into the X register and "a" into the A register. The vector is rotated in successive steps by choosing the sign of k such that the contents

*To illustrate, consider the binary equivalent of the decimal number ten, i.e. 1010, to be stored in a shift register. Shifting the bits to the right one register position will cause the binary 0101 (decimal 5) to appear in the register which is equivalent to multiplying the original number by 1/2.

of the A register is reduced to within 1 bit of zero in 16 steps. In the process of rotation, a computation corresponding to the inverse of that for the cartesian-to-polar resolution is performed, with the X component of the original vector being formed in the X register and Y in the Y register.

GENERAL DESCRIPTION OF PROGRAM OPERATION

A schematic diagram of the DOS Resolver program is shown in Figure 2. Serial 2's complement 13-bit binary coded numbers, representing sampled values of variables x, y, and a, are accepted at a 2 m.c. bit-rate from the buffered output of the A/D converter during the first 8 microseconds of the 128 microsecond computing cycle. Trigonometric transformations are performed on sets of input numbers in a sequential fashion, although these operations need not be in the same mode or involve the same variables. Computation accuracy is 1 bit in 12, which is equivalent to an accuracy of 0.025%. Although the basic resolver program is capable of performing up to four (4) independent transformations, it can be expanded, when desired, to more than four, with the maximum limited by the 20 channel capacity of the multiplexer. The 16 steps of each rotation or conversion cycle takes only 16 DOS "word times" or 128 microseconds. The sampling rate can be calculated from the expression

$$\text{Samples per second} = \frac{1}{250 \times 10^{-6} \times \text{No. of inputs} \times \text{No. of resolvers}}$$

The program can also be expanded conveniently to provide additional control and storage of temporary results, using the flexible logic of the DOS. Also, the X and Y registers can be duplicated, which corresponds to having extra sine-cosine cards on an analog resolver.

Operation of the basic program functions can be described briefly as follows:

Mode Control: The computation mode and the number of inputs accepted by a resolver are selected by the states of switched binary signals. A binary ZERO selects the 0 mode of operation; a binary ONE the 1 mode. If the resolver is to accept two (2) inputs, a binary ZERO is used: three (3) inputs requires a binary ONE. In the basic program, these switched variables are provided by Function Switches according to the schedule presented in Table I. They can be supplied also from flip-flops set by control logic patched on the DOS, or even from binary reference signals terminated on the DOS patch panel.

The number of resolvers being used is set on a Pre-Set Switch whose output is fed into a two-stage counter circuit. The output of the counter, which automatically resets at the end of the count set on the Pre-Set Switch, is gated with the mode-selector signals and the number-of-inputs signals to produce a signal which determines whether the sign of A or the sign of Y is controlling the computation for a particular resolver. When the sign of A is controlling, the angle is being reduced to zero; if the sign of Y controls, the Y is being zeroed out.

Input Cycle: Analog variables serving as inputs to the resolvers are sequentially assigned to multiplexer channels, beginning with channel #1, until all inputs have been assigned. Within each resolver group, the order of the inputs is as follows:

1st - x input.

2nd - y input, or A input if the resolver is being used in the 0 mode with two inputs.

3rd - A input if three inputs are being used.

The number of channels scanned by the multiplexer before it resets must be equal to the total number of inputs to all resolvers. For example, if three resolvers are being used, two with 2 inputs and one with 3 inputs, the multiplexer must be programmed to scan 2 + 2 + 3 = 7 inputs before resetting.

The output of the multiplexer feeds into an A/D converter which converts the sampled analog variables into 13-bit (12 bits + sign) binary numbers. The multiplexer-converter combination cycles continuously; a conversion FINISH signal increments the multiplexer to the next channel, switching a new sampled variable into the converter for conversion, etc.. A down-counter circuit stepped by the converter FINISH signal causes the first binary number converted in an input cycle to be loaded into a Memory Buffer (1 word storage); the second word into a second Memory Buffer. At this point, the counter circuit makes a decision, based upon the number of input signals, as to the length of the input cycle. If the resolver is to receive only two inputs, then the input cycle begins again. If three inputs are required, then the third converted binary word is loaded into a third Memory Buffer, after which the input cycle begins again. At the end of the input cycle, FF 300 is set to the high state to start the computation cycle, at which time the information stored in the Memory Buffers is gated into the proper registers.

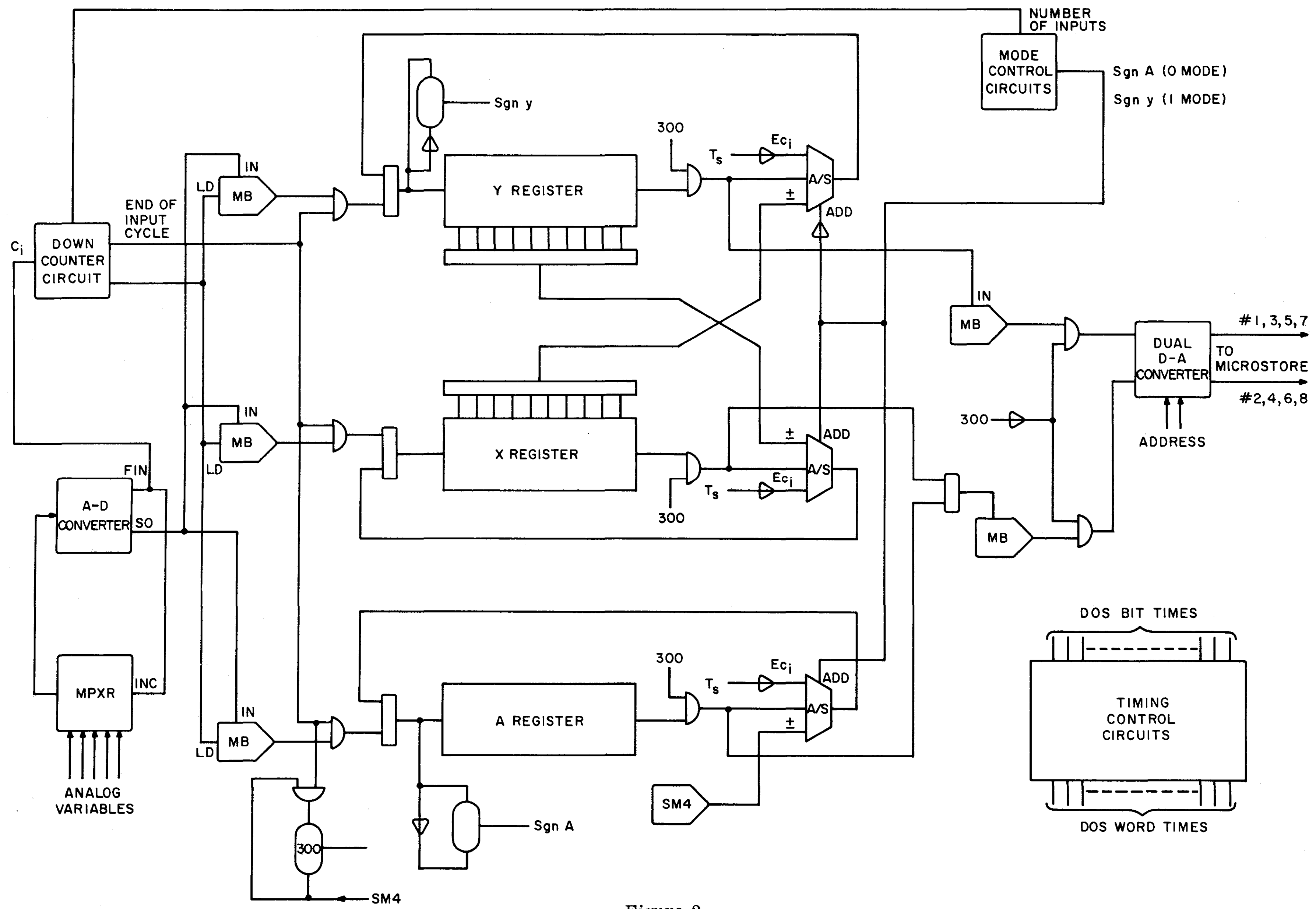

Figure 2
Block Diagram of DOS Resolver Program

Table I
Mode and Number of Inputs Switch Assignments

Function Switch Number	Function of Switch
00	Number of Inputs for Resolver # 1
01	Number of Inputs for Resolver # 2
10	Number of Inputs for Resolver # 3
11	Number of Inputs for Resolver # 4
20	Mode of Resolver # 1
21	Mode of Resolver # 2
30	Mode of Resolver # 3
31	Mode of Resolver # 4

Timing Control: During the first 8 microseconds (1 DOS word time) of the computation cycle, the registers are loaded. During the next word time, the vector is rotated 90 degrees. At this point, the iteration process used in performing the computation begins, one iteration per word-time being performed. Word-timing signals generated in the DOS are used to control the values of k such that $k_i = k_{i-1}/2$, and the gating of kx and ky out of the registers. At the end of the computation cycle, FF 300 is reset to the low state by an SM-4 signal (the last bit of the last SM-4 word). The time required for the various operations, and the order of their sequence, are shown in the Timing Diagram of Table II.

Computation Cycle: The X and Y registers each have an Adder/Subtractor Unit which receives two inputs: the gated output of a register, and a signal which represents the product of k times the contents of the other register. This second signal is added to or subtracted from the first according to a signal from the mode control circuit. There is a third Adder/Subtractor Unit, also with two inputs: the output of the A register, and a signal representing the arctangent of 2^{-i} as stored in the DOS SM-4 Serial Memory. This second signal is added to or subtracted from the first signal again according to a control signal from the mode control circuit. The angles stored in the SM-4 are shown in Table III.

Output Cycle: During the last word-time of the computation cycle, the two outputs forming the results of the computation are loaded into Memory Buffers. FF 300 then goes low, signaling the end of the computation cycle, and the contents of the Memory Buffers are loaded into the properly addressed dual channels of the D/A Converter, one word-time apart. The D/A conversion then begins; 96 microseconds later, the MICROSTORE unit associated with the proper resolver channel (see Table IV) is placed in the "Track" mode to accept the analog voltage comprising the outputs of the converter; 400 microseconds later, the MICROSTORE units are switched back to the "Store" mode and the output cycle is ready to perform the output cycle for the next resolver channel.

SCALING OF VARIABLES

The analog variables serving as inputs to the DOS Resolver program must be scaled such that the magnitude of each individual input is equal to or less than 100 volts, thus

$$\left|\frac{x}{K_R}\right| \leq 100 \text{ volts}$$

$$\left|\frac{y}{K_R}\right| \leq 100 \text{ volts}$$

$$\left|\frac{R}{K_R}\right| \leq 100 \text{ volts}$$

where $K_R = 0.82338$.

Angle inputs must be scaled such that

$$180° = 100 \text{ volts}$$

or

$$K_B = 0.55555.$$

The unscaled output also must be less than 100 volts. The scaled output then will be

$K_R X$,

$K_R Y$, or

$K_R R$

Table II
Timing Diagram for DOS Resolver Program

TIME INTERVAL μ SEC	INPUT	COMPUTATION	OUTPUT
128	1st INPUT CONVERSION RESOLVER #1		
128			
128	2nd INPUT CONVERSION RESOLVER #1		
128			
128	1st INPUT CONVERSION RESOLVER #2	COMPUTATION RESOLVER # 1	
128			OUTPUT CONVERSION RESOLVER #1
128	2nd INPUT CONVERSION RESOLVER #2		
128			OUTPUT CONVERSION RESOLVER # 1
128	1st INPUT CONVERSION RESOLVER #3	COMPUTATION RESOLVER # 2	
128			OUTPUT CONVERSION RESOLVER # 2
128	2nd INPUT CONVERSION RESOLVER #3		
128			OUTPUT CONVERSION RESOLVER # 2
128	3rd INPUT CONVERSION RESOLVER #3		
128			
128	1st INPUT CONVERSION RESOLVER #1	COMPUTATION RESOLVER # 3	
128			OUTPUT CONVERSION RESOLVER # 3

Table III
Angles Stored in SM-4 Serial Memory

90°
45°
26.5650°
14.0362°
7.1250°
3.5767°
1.7898°
0.8947°
0.4467°
0.2293°
0.1115°
0.0558°
0.0281°
0.0138°
0.0068°

Table IV
MICROSTORE Assignments for Outputs

Track-Hold Amplifier Number	Output
1	X of Resolver # 1
2	Y of Resolver # 1 (0 Mode)
2	A of Resolver # 1 (1 Mode)
3	X of Resolver # 2
4	Y of Resolver # 2 (0 Mode)
4	A of Resolver # 2 (1 Mode)
5	X of Resolver # 3
6	Y of Resolver # 3 (0 Mode)
6	A of Resolver # 3 (1 Mode)
7	X of Resolver # 4
8	Y of Resolver # 4 (0 Mode)
8	A of Resolver # 4 (1 Mode)

APPENDIX I

The complement of HYDAC components required to implement the DOS Resolver program can be summarized as follows:

Components	Number Required
Track-Hold Amplifier (MICROSTORE)	2 per resolver
Multiplexer	1
Analog-to-Digital Converter	1
Digital-to-Analog Converter (Dual)	1
AND Gates	120
General Purpose Flip-Flops	22
SM-4 Serial Memory Unit	1
Memory Buffer	3
Quad Shift Register	12
Digital Function Switch	2 per resolver
Pre-Set Switch	1
Adder/Subtractor Unit	3