intel®

AP-406

APPLICATION NOTE

MCS®-96 Analog Acquisition Primer

DAVID P. RYAN
INTEL CORPORATION

December 1987

Order Number: 270365-001

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

ANALOG ACQUISITION PRIMER

CONTENTS

CONTENTS

PAGE

LISTING OF FIGURES

CONTENTS

PAGE

LISTINGS

THE MCS®-96 ANALOG ACQUISITION PRIMER

INTRODUCTION

As technology advances, embedded control applications continue to reduce chip-count and demand microcontrollers with increased features to assist system-cost reduction. Since every embedded control application interfaces with the physical world, and the physical world is an analog process, it was inevitable that microcontrollers would include integrated analog acquisition capabilities.

The first such integration of standard microcontroller and A/D converter occurred on Intel's 8022 in 1978. This opened the door to cost reduction of high volume applications that required analog inputs. The device fit well into applications that needed processing of analog data. But this chip, with its 8-bit CPU, could not perform in high-end applications requiring analog inputs, or in applications that had computationally demanding analog tasks.

With the introduction of the MCS®-96 family of 16-bit microcontrollers in 1982, the combined CPU and A/D performance became available to greatly reduce the system cost of mid- and high-performance embedded control applications. These are applications which were customarily implemented with 16-bit microprocessor chip-sets teamed with analog acquisition chip sets.

There are less obvious avenues for system cost reduction when a 16-bit CPU is teamed with an on-chip analog acquisition system. For example, closed-loop servo control had been implemented almost exclusively by using analog methods. When an MCS-96 device is designed into such an application, it is not only replacing a microcontroller or microprocessor, but it also replaces closed-loop analog circuitry which never before came in contact with the digital system.

To take full advantage of this new level of integration, digital designers must become familiar with analog acquisition, and analog designers must become familiar with digital methods of processing analog signals. This Application Note assists with the first task—understanding of an analog acquisition system.

Designers experienced with analog design, or analog acquisition systems, may find no revelations herein. To those unfamiliar with analog acquisition systems, this Ap Note provides a tutorial on the subject and will serve as a handy reference.

Answering the limitless number of analog circuit design questions is beyond the scope of this Ap Note. Suffice it to say that the effort placed on the design of analog circuits should increase with a decreasing error budget.

At a minimum, the applications literature of op-amp manufacturers and analog design manuals are a good place to start. Furthermore, the applications literature of monolithic analog acquisition system manufacturers should be consulted since the suggestions presented therein are largely transportable to any A/D system.

This Ap Note is organized in the following sections. The components of an analog acquisition system and the errors associated with each is first explained. Then, interfacing suggestions and ideas for getting more resolution are presented. Finally, a set of appendices provides back-up information, a bibliography, actual converter data and some program listings.

The definitions of terms used, and the examples presented, are drawn from the body of applications literature publicly available on the components of an analog acquisition system. There is usually no single meaning for a particular term or specification used to describe analog acquisition. However, there is, in most cases, a generally accepted definition which is most often used. To the extent possible, we have adopted the most used definition. To avoid any ambiguity, Appendix A lists the dictionary of terms as used to refer to the analog acquisition systems of MCS-96 devices.

For any users of an MCS-96 analog acquisition system (experienced or not), this document contains very useful information. It should be considered mandatory reading in addition to the latest Embedded Controller Handbook and MCS-96 data sheet for the actual device in use prior to the actual design.

WHAT IS AN ANALOG ACQUISITION SYSTEM?

An analog acquisition system is a collection of individual units which, when logically configured, form a system capable of converting an analog input to a digital value.

The typical components of an Analog Acquisition Unit (Figure 1) include an Analog-to-Digital Converter (A/D), a Sample-and-Hold (S/H) and an Analog Multiplexer (MUX). The A/D converts the infinitely varying analog voltage present on the S/H into a digital representation for use by the digital system. The S/H is required so a "snapshot" of a changing analog input can be stored for conversion by the A/D. The MUX is used to leverage the investment in the A/D by allowing a large number of isolated analog input channels to use the same converter.

The conversion result of an MCS-96 device is a 10-bit ratiometric representation of the input voltage. This produces a stair-stepped transfer function when the output code is plotted versus input voltage. See Figure 2.

The resulting digital codes can be taken as simple ratiometric information, or they can be used to provide information about absolute voltages or relative voltage changes on the inputs. The more demanding the application is on the A/D converter, the more important it is to fully understand the converter's operation. For simple applications, knowing the absolute error of the converter is sufficient. However, controlling a closed loop with analog inputs necessitates a detailed understanding of an A/D converter's operation and errors.

The errors inherent in an analog-to-digital conversion process are many: quantizing error; zero offset; full-scale error; differential non-linearity; and non-linearity. These are "transfer function" errors related to the A/D converter. In addition, the S/H and MUX may induce channel dissimilarities and sampling error (described later).

Fortunately, one "Absolute Error" specification is available which describes the sum total of all deviations between the actual conversion process and an ideal converter. The various sub-components of error are, however, important in many applications. These error components are described in Appendix A and in the text below where ideal and actual converters are compared.

A/D Converter

There are at least three well-recognized methods for converting an analog voltage to a digital value—flash, dual slope and successive approximation.

Flash A/Ds are the fastest, and most expensive converters for a given accuracy. Flash converters typically resolve bits of the result in parallel to achieve fast conversions. Flash converter speeds are measured in tens-of-nanoseconds.

Dual slope converters are the slowest, but most accurate. Dual slope conversion is rather insensitive to noise on the input, but conversion times are measured in milliseconds.

Successive approximation converters provide a balanced tradeoff between speed and accuracy. Successive approximation conversion times are measured in tens-of-microseconds, and converter implementations are very economical for a given accuracy.

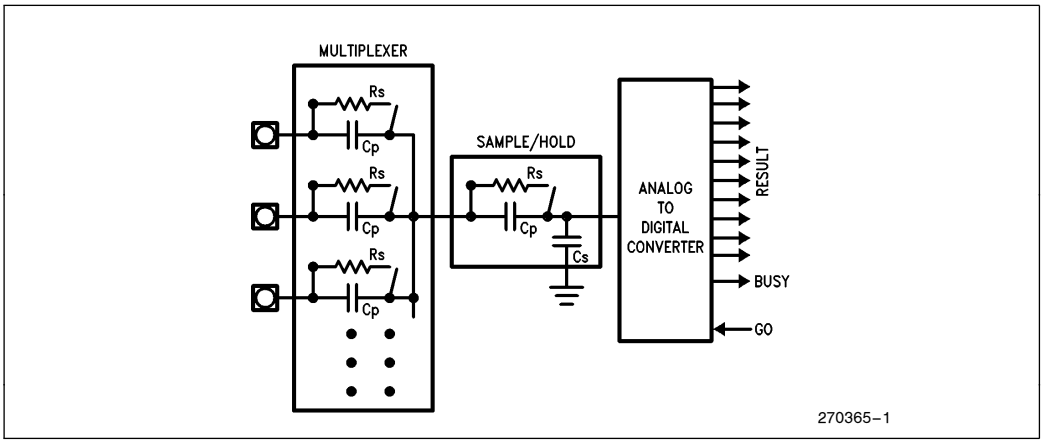


Figure 1. An Analog Acquisition System

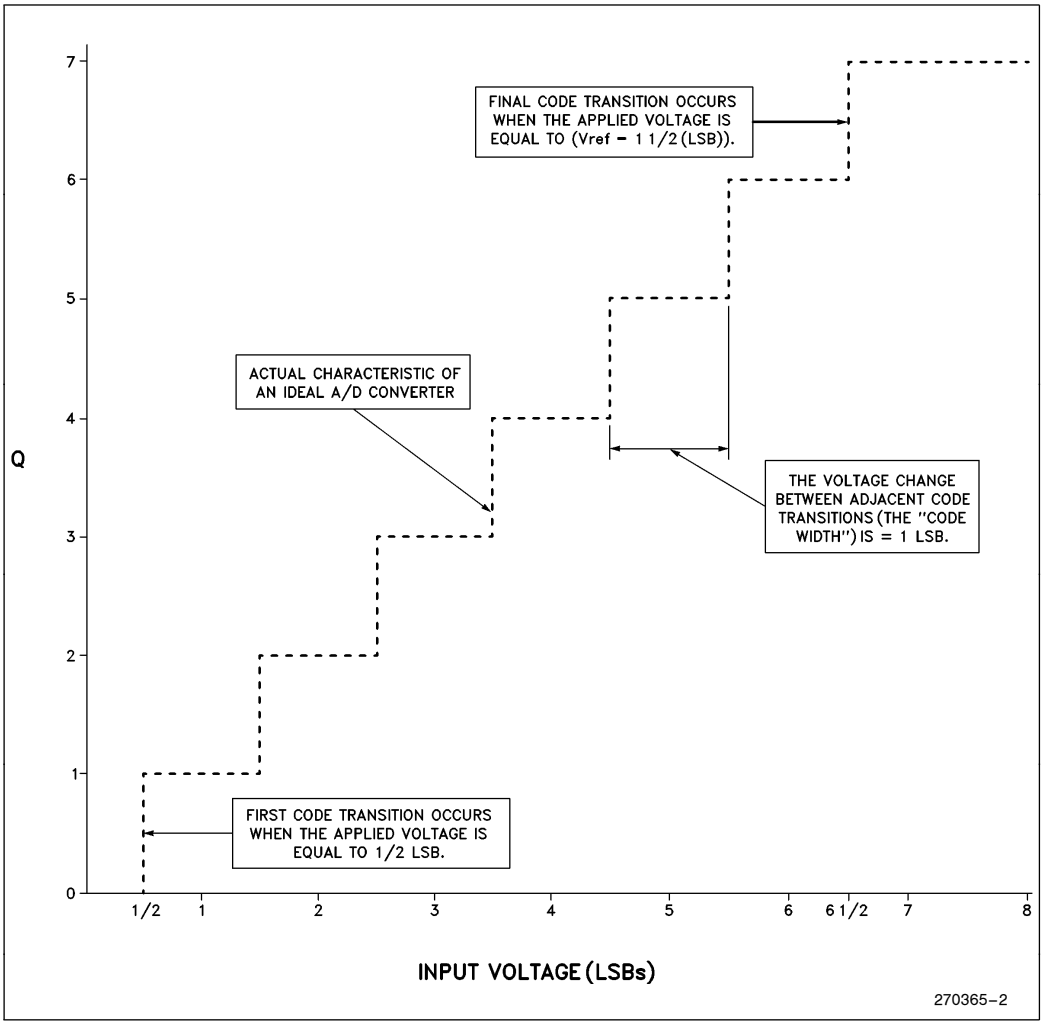


Figure 2. Ideal A/D Characteristic

MCS-96 converters use successive approximation. A successive approximation conversion is performed by comparing a sequence of reference voltages to the analog input in a binary search for the reference voltage that most closely matches the input. The ½ full-scale reference voltages is the tested first. This corresponds to a 10-bit result where the most significant bit is zero, and all other bits are ones (0111 1111 11b). If the analog input is less than the test voltage, bit 10 of the result is left a zero, and a new test voltage of ¼ full-scale (0011 1111 11b) is tried. If this test voltage is lower than the analog input, bit 9 of the result is set and bit 8 is cleared for the next test (0101 1111 11b). This binary search continues until 10 tests have occurred, at which time the valid 10-bit conversion result resides in a register where it can be read by software.

The voltages used during the binary search are generated from an internal Digital-to-Analog Converter similar to Figure 3. The figure shows eight resistors being used as a three-bit D to A. The first resistor tap is taken from the center of the first resistor to guarantee that a zero input voltage will always output a zero code. Each successive tap then provides a reference voltage $V_{REF}/8$ (one LSB) from the previous tap. When the analog input is above the voltage of the seventh tap, the A/D will resolve to its full-scale value of 111b. Therefore, an eighth tap is not needed, and the A/D's 110b to 111b code transition will occur when V_{ANIN} equals $V_{REF} - 1\frac{1}{2}$ LSB.

The first error seen in this process is unavoidable, and results from the conversion of a continuous voltage to

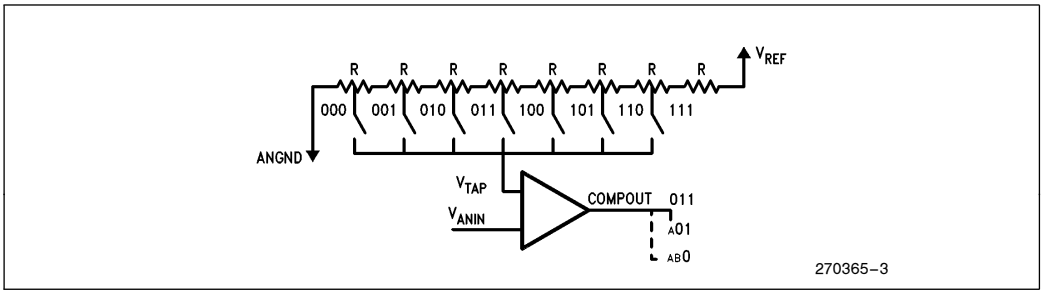


Figure 3. A Three-Bit D-to-A

an integer digital representation. This error is called quantizing error, and is always ±0.5 LSB. Quantizing error is the only error seen in a perfect A/D converter, and is obviously present in actual converters. Figure 2 shows the transfer function for an ideal 3-bit A/D converter (i.e. the Ideal Characteristic).

Note that in Figure 2 the Ideal Characteristic possesses unique qualities: it's first code transition occurs when the input voltage is 0.5 LSB; it's full-scale code transition occurs when the input voltage equals the full-scale reference minus 1.5 LSB; and it's code widths are all exactly one LSB. These qualities result in a digitization without offset, full-scale or linearity errors. In other words, a perfect conversion.

Figure 4 shows an Actual Characteristic of a hypothetical 3-bit converter which is not perfect. When the Ideal Characteristic is overlaid with the imperfect characteristic, the actual converter is seen to exhibit errors in the location of the first and final code transitions and code widths. The deviation of the first code transition from ideal is called "zero offset". The deviation of the final code transition from ideal is "full-scale error".

The deviation of the code widths from ideal causes two types of errors. Differential Non-Linearity and Non-Linearity. Differential Non- Linearity is a local linearity error measure, whereas Non-Linearity is an overall linearity error measure. For example, Figure 5a shows a transfer function with a large differential non-linearity and a little non-linearity. In contrast, Figure 5b shows a characteristic with small differential errors but a large overall linearity error.

Differential Non-Linearity is the degree to which actual code widths differ from the ideal width. Differential Non-Linearity gives the user a measure of how much the input voltage may have changed in order to produce a one count change in the conversion result.

If the absolute value of an input voltage is less important than the amount that the input changes, the differential non-linearity (DNL) specification of a converter is very important. For example, if the differential non-linearity of a converter is less than ± 0.5 LSB, a one count change in the digital result means that the input voltage changed at most 1.5 LSB (1 LSB ideal ± 0.5 LSB DNL). This is a much more accurate description of the input voltage change than would be available if the differential non-linearity of the converter was not known.

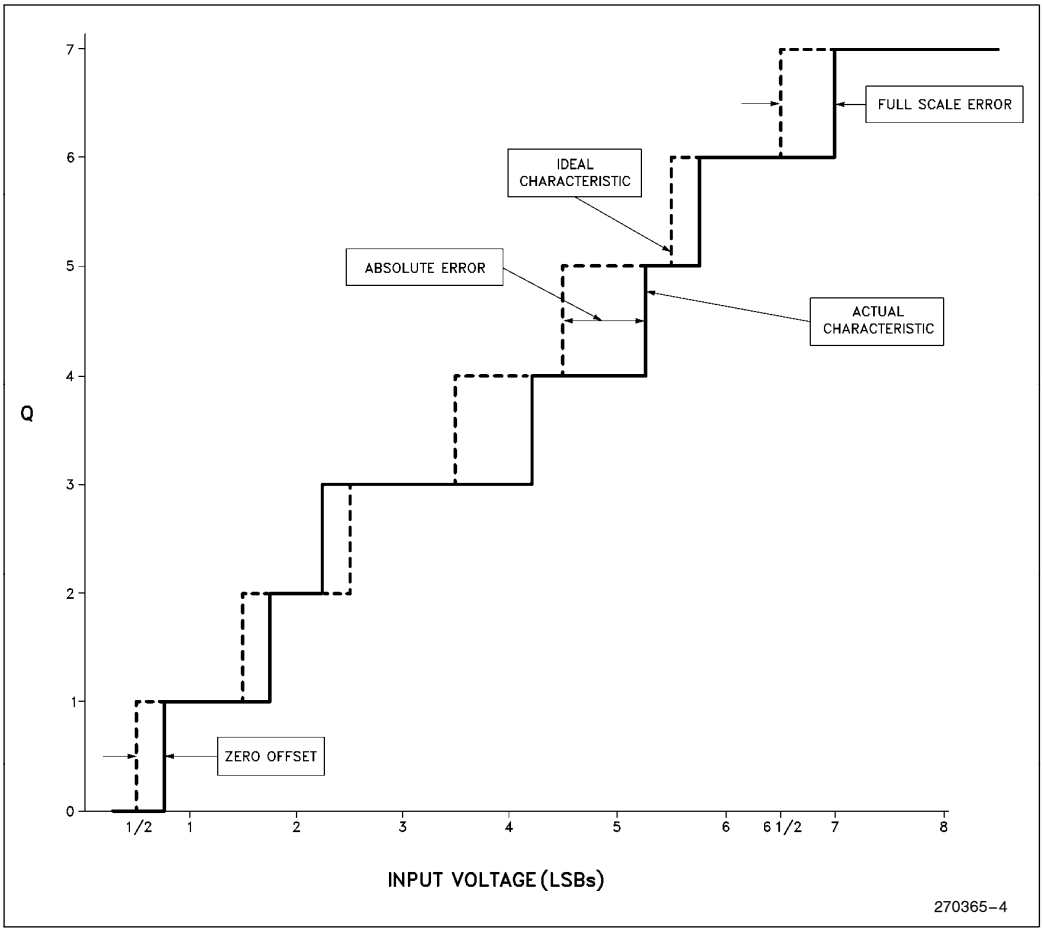


Figure 4. Actual and Ideal Characteristics

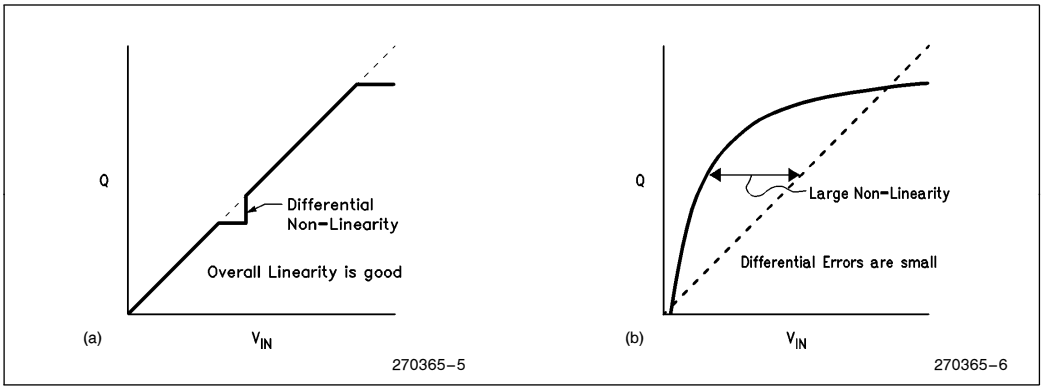


Figure 5. Types of Linearity Errors

Non-Linearity is the worst case deviation of code transitions from the corresponding code transitions of the Ideal Characteristic. Non- Linearity describes how much Differential Non-Linearities could add to produce an overall maximum departure from a linear characteristic.

If the Differential Non-Linearity errors are large enough, it is possible for an A/D converter to miss codes or exhibit non-monotonicity. Neither behavior is desirable in a closed-loop system. A converter has no missed codes if there exists for each output code a unique input voltage range that produces that code only. A converter is monotonic if every subsequent code change represents an input voltage change in the same direction. Figure 6a shows a converter with missed codes. Figure 6b shows a non-monotonic converter.

Differential Non-Linearity and Non-Linearity are quantified by measuring the Terminal Based Linearity Errors. A Terminal Based Characteristic results when an Actual Characteristic is shifted and scaled to eliminate zero offset and full-scale error (see Figure 7). The Terminal Based Characteristic is similar to the Actual Characteristic that would be seen if zero offset and full-scale error were externally trimmed away. In practice, this is done by using input circuits which include gain and offset trimming. (See the Application Hints section for more details.)

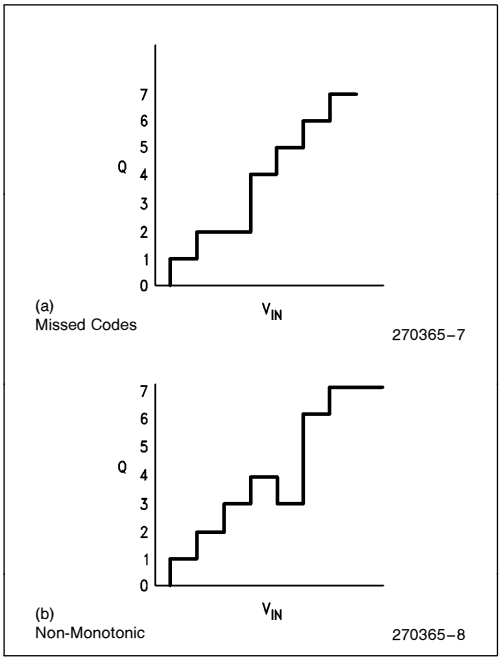


Figure 6. Undesirable Converter Operation

An often overlooked characteristic of A/D converters is that code transitions do not really occur instantaneously at some finite set of input voltages. Specific code transitions can be analyzed by doing repeated conversions around the transition point using a high accuracy input voltage. When this is done, we find that there is actually a range of voltages around code transitions where both the lower and upper codes occur for repeated conversions on the same input voltage.

Figure 8 shows this "repeatability" error. At the lower end of the region of repeatability error the lower code is most prevalent, but the upper code will occur in a small percentage of the conversion attempts. As the input voltage increases slightly, a point is reached where both lower and upper codes occur with 50 percent probability. As the input voltage moves slightly higher, the upper code occurs most often with the lower code showing up in a small percentage of conversions.

The repeatability error is due to the fundamental ability of the comparator in the A/D to resolve very similar voltages. Random noise also contributes to repeatability errors. On MCS-96 devices, the width of the region of repeatability error has been found to be typically 1 mV to 1.25 mV. Since this error is specified, all other errors are specified assuming the code transitions occur at the voltage where adjacent codes are equally likely.

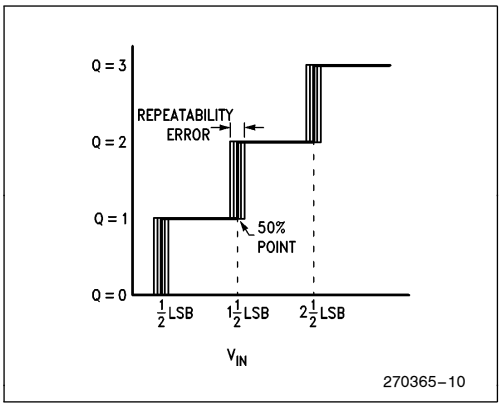


Figure 8. Repeatability Error

The Multiplexer

The eight channel multiplexer is implemented as a collection of eight MOS switches. Only one of eight can be closed at any instant in time. Figure 1 shows the multiplexer with the switches acting as resistors when closed and as small parasitic capacitors when open. The input protection devices on the analog input pins are also considered a part of the multiplexer.

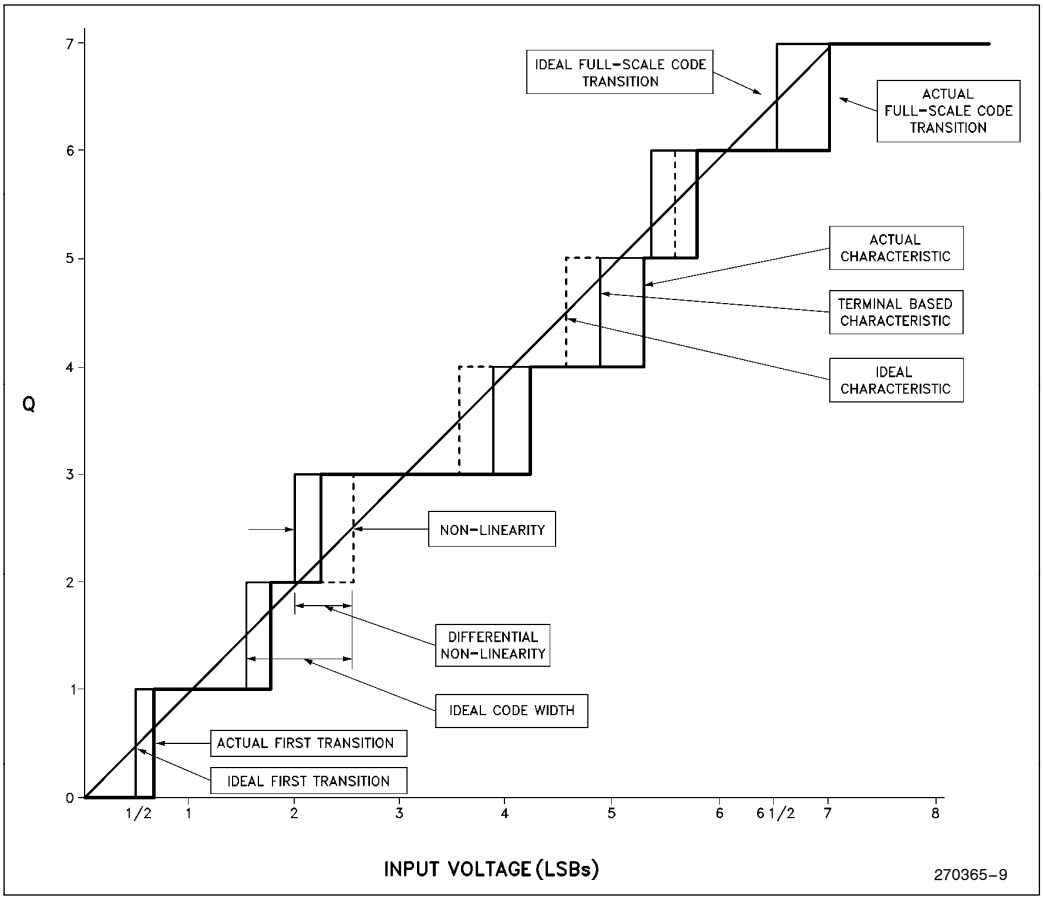


Figure 7. Terminal Based Characteristic

The resistance of a closed switch is typically 1K to 2K ohms and the D.C. leakage due to the input protection is typically 3 microamps maximum. Both values depend upon the process used and day-to-day fabrication variations. The channel resistance and the D.C. leakage can also vary from channel-to-channel on the same device. These variations can be seen in the conversion process and are described by the channel- to-channel matching specification.

Channel-to-channel matching specifies the input voltage differences induced by mismatched elements of the multiplexer. This error is quantified by measuring the difference between the input voltages necessary to cause the same code transition to occur through different multiplexer channels under identical test conditions.

Matching errors are more complex than a simple voltage offset between channels, and thus are difficult to externally cancel. Fortunately, multiplexer channels typically match to within one millivolt.

A multiplexer that has the potential to short two inputs together is not very attractive. To keep this from happening, the circuitry that selects the active channel is designed to guarantee that all channels are deselected before a new channel is selected. Thus, the multiplexer is said to be Break-Before-Make.

In addition to Break-Before-Make channel selection, an analog multiplexer must be able to keep deselected channels isolated from the selected channel. As shown in Figure 1, there are parasitic capacitances coupling every deselected channel to the multiplexer output. The quantification of coupling is called Off-Isolation. Off-isolation is the multiplexer's ability to attenuate signals on deselected channels.

Sample-and-Hold

The sample-and-hold of an analog acquisition system can be built using an analog switch and a sample capacitor. As with the multiplexer, there is also a parasitic capacitance coupling the switch input to the sample capacitor when the switch is open (Figure 1).

The resistance of the sample-and-hold switch combines with the series resistance of the multiplexer to impede the current necessary to charge the sample capacitor. For example, with a 5K ohm total input resistance from the pin to the 2 pf sample capacitor, the RC time constant is 10 nS (2 pf × 5K ohms).

During the one microsecond that the sample capacitor is connected to the input, 100 time constants elapse (1 microsecond/10 nS). This means that the sample capacitor is 100 percent of the voltage on the input pin ($1\text{-}e^{-100}$), assuming a zero source impedance.

If a source impedance of 2K ohms is assumed, the RC time constant of the sampling process would be 14nS (7K ohms × 2 pf). Thus, 71.4 time constants would pass in one microsecond resulting in the sample capacitor being charged to within 99.9 percent of its final value. Source impedances above 2K ohms would begin to degrade the conversion accuracy due to D.C. leakage (described later).

Figure 9 shows the actual input voltage and the sampled voltage approaching the input voltage. Once the sample-and-hold switch closes, the sample window begins. The sample window extends for four state times and ends with the sample-and-hold switch opening on MCS-96 devices (except 8X9X-90, which is 8 state times and has no sample-hold). Figure 9 also shows the sample delay, which is the delay from the time a start conversion signal is generated to the time a conversion process begins.

It is important to understand the uncertainties associated with the timing of the sample-and-hold. Digital signal processing algorithms rely upon the "spectral purity" of the sampling process. If the sample window jumps around with respect to the start conversion signal, or if the start conversion signal cannot be generated at precise times, consecutive samples of input data will not be equally spaced in time (i.e. sampling will be spectrally impure).

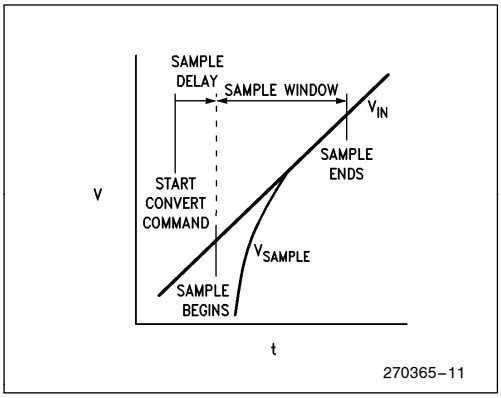


Figure 9. Sample-and-Hold Voltage

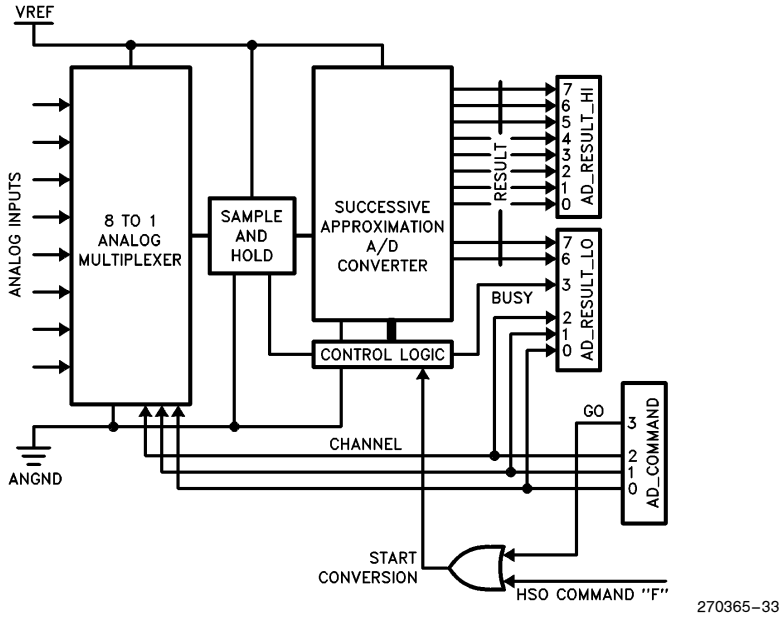
To improve the spectral purity of the sampling in digital signal processing applications, sequential MCS-96 start conversion signals can be generated with less than 50 nanoseconds of jitter using the HSO unit. The sample delay and sample time are also a constant number of state times to within 50 nanoseconds each.

Once the sample window closes, it is desired that all further changes on any input channel be isolated from the sample capacitor. The multiplexer's off-isolation is responsible for isolating deselected channels, while the sample-and-hold switch must attenuate changes on the selected channel. This source of error is described as Feedthrough. Feedthrough is quantified as the ability of the sample-and-hold to reject unwanted signals on its input.

Other factors that affect a real A/D Converter system include sensitivity to temperature. Temperature sensitivities are described by the change in typical specifications with a change in temperature.

The MCS®-96 Conversion Sequence

The MCS-96 Analog Acquisition System includes an eight channel analog multiplexer, sample-and-hold circuit and 10-bit analog to digital converter (Figure 10). An MCS-96 device can therefore select one of eight analog inputs, sample-and-hold the input voltage and convert the voltage into a digital value. Each conversion takes 22 microseconds (8097BH), including the time required for the sample-hold (with XTAL1 = 12 MHz). The method of conversion is successive approximation.



NOTE:
1. Sample and hold not on 8X9X-90 devices.

Figure 10. A/D Converter Block Diagram

The conversion process is initiated by the execution of HSO command OFH, or by writing a one to the GO Bit in the A/D Control Register. Either activity causes a start conversion signal to be sent to A/D control logic. If an HSO command was used, the conversion process will begin when Timer 1 increments. This aids applications attempting to approach spectrally pure sampling, since successive samples spaced by equal Timer 1 delays will occur with a variance of about ±50 ns (assuming a stable clock on XTAL1). However, conversion initiated by writing a one to the ADCON register GO Bit will start within three state times after the instruction has completed execution, resulting in a variance of about 0.75 μs (XTAL1 = 12 MHz).

Once the A/D unit receives a start conversion signal, there is a one state time delay before sampling (sample delay) while the successive approximation register is reset and the proper multiplexer channel is selected. After the sample delay, the multiplexer output is connected to the sample capacitor and remains connected for four state times (sample time). After this four state time "sample window" closes, the input to the sample capacitor is disconnected from the multiplexer so that changes on the input pin will not alter the stored charge while the conversion is in progress. The sample delay and sample time uncertainties are each approximately ±50 ns, independent of clock speed.

To perform the actual analog-to-digital conversion the MCS-96 implements a successive approximation algorigthm. The converter hardware consists of a 256-resistor ladder, a comparator, coupling capacitors and a 10-bit successive approximation register (SAR) with logic that guides the process. The resistor ladder provides 20 mV steps (V_{REF} = 5.12V), while capacitive coupling is used to create 5 mV steps within the 20 mV ladder voltages. Therefore, 1024 internal reference voltages are available for comparison against the analog input to generate a 10- bit conversion result. Appendix B contains a detailed description of the method used to generate 1024 voltages from a 256-resistor chain.

The total number of state times required for a 10-bit conversion varies from one MCS-96 version to the next. Attempting to short-cycle the 10-bit conversion process by reading A/D results before the done bit is set may work on some versions of MCS-96 devices, however it is not recommended. Short-cycling is not tested, nor is it guaranteed. Furthermore, it may not work on future MCS-96 devices.

APPLICATION HINTS

The analog signals that must be converted by an analog acquisition system vary widely. The analog input may arrive at the controller as a voltage or current. The range may be 0 to 1 volt or ±30 volts, or some other arbitrary range. The input may be linear, logarithmic, non- linear, or perturbated in some bizarre fashion. Although interfacing to such signals could be considered an art form, some simple suggestions are contained in this section.

Analog Inputs

The external interface circuitry to an analog input is highly dependent upon the application, and can impact converter characteristics. In the external circuit's design, important factors such as input pin leakage, sample capacitor size and multiplexer series resistance from the input pin to the sample capacitor must be considered.

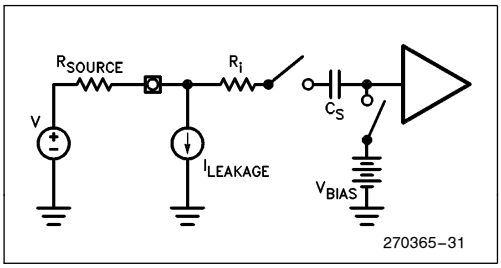


Figure 11. Idealized A/D Sampling Circuitry

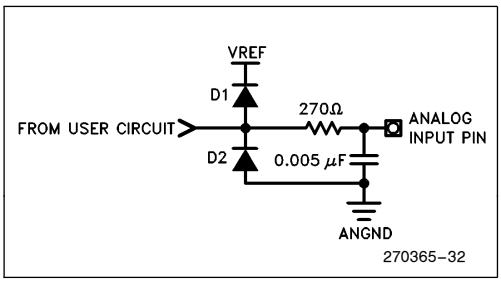


Figure 12. Suggested A/D Input Circuit

For the 8096BH, these factors are idealized in Figure 11. The external input circuit must be able to charge a sample capacitor (C_S) through a series of resistance (R_I) to an accurate voltage given a D.C. leakage (I_L). On the 8096BH, C_S is around 2 pF, R_I is around 5 KΩ and I_L is specified at 3 μA maximum. In determining the source impedance R_S, V_{BIAS} is not important.

External circuits with source impedances of 1 KΩ or less will be able to maintain an input voltage within a tolerance of about ±0.61 LSB (1.0 KΩ × 3.0 μA = 3.0 mV) given the D.C. leakage. Source impedances above 2 KΩ can result in an external error of at least one LSB due to the voltage drop caused by the 3 μA leakage. In addition, source impedances above 25 KΩ may degrade converter accuracy as a result of the internal sample capacitor not being fully charged during the 1 μs (12 MHz clock) sample window.

Placing an external capacitor on each analog input will reduce the sensitivity to noise, as the capacitor combines with source resistance in the external circuit to form a low-pass filter. In practice, one should include a small series resistance prior to an external low leakage capacitor on the analog input pin and choose the largest capacitor value practical, given the frequency of the signal being converted. This provides a low-pass filter on the input, while the resistor will also limit input current during over-voltage conditions.

Figure 12 shows a simple analog interface circuit based upon the discussion above. The circuit in the figure also provides limited protection against over-voltage conditions on the analog input (limits to 2.6 mA with 270Ω (0.7/270)). The circuit induces leakage from the diodes, which should be kept small.

The wide range of possible analog environments that must be interfaced to, or the existence of stringent accuracy requirements, makes the consideration of alternative input buffer configurations necessary. The most popular input buffer is a single op-amp in the non- inverting or inverting configurations of Figure 13.

In the non-inverting circuit of Figure 13 (a), the analog input is scaled by the buffer gain to output 5 volts when the input is at its maximum positive input. When the buffer input is 0 volts, the output will also be 0 volts.

In the inverting circuit of Figure 13 (b), a reference equal to the maximum possible input voltage is placed on the non-inverting input of the op-amp and the actual analog input is placed on the inverting input. The output voltage of the buffer is then proportional to the deviation of analog input from its maximum possible value. For example, when the analog input equals V_{MAX}, the buffer output will equal 0 zero volts. When the analog input equals its minimum value, the buffer output equals 5 volts. The digital result from the A/D converter might, of course, have to be complemented before being used.

The circuits of Figure 13 show only feedback resistors that set the gain of the buffer. In practice, it will often be necessary to include offset adjustments, gain trimming, temperature or frequency stability compensation, or components to build an active filter.

Figure 14 depicts a generalized non-inverting input buffer that offsets the analog input and scales the input

to a 5 volt range. The course offset is set by the ratio of R_{BIG1} and R_{BIG2}, while offset fine tuning is done by adjusting R_{TRIM}. The course gain is set by the ratio of R_{G1} and R_{G2} while gain trimming is done with R_{GTRIM}.

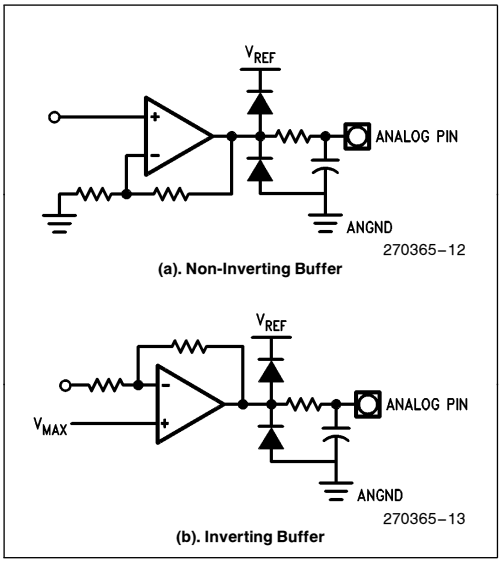


Figure 13

By trimming the offset and gain, not only can external component errors be zeroed out, but the offset and full scale error of the A/D converter can be nulled.

The procedure for nulling offset and gain is simple. First, a voltage is applied to V_{IN} which corresponds to the ideal first code transition of the A/D. R_{TRIM} is adjusted so that 50 percent of the conversion results are 0 while 50 percent are 1. Second, a voltage is applied to V_{IN} which corresponds to the ideal final code transition of the A/D converter. R_{GTRIM} is then adjusted until 50 percent of the conversion results are 3FEH and 50 percent are 3FFH. Once this adjustment is complete, the converter zero offset and full-scale errors are nulled, and could be ignored (except for temperature variation). This allows the system to rely upon the tighter, more descriptive converter specifications for Terminal Based Non- Linearity and Differential Non-Linearity.

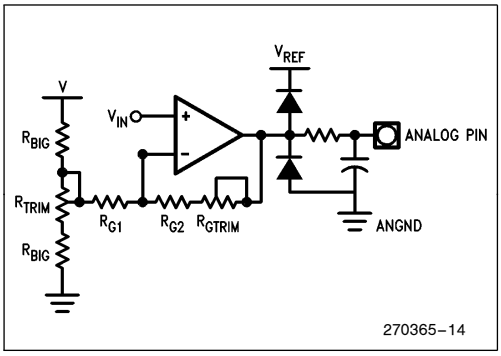


Figure 14. Trimming Offset and Gain

Analog References

Reference supply levels strongly influence the absolute accuracy of the conversion. For this reason, it is recommended that the ANGND pin be tied to a clean ground, as close to the power supply as possible. Bypass capacitors should also be used between V_{REF} and ANGND. ANGND should be connected to V_{SS} only at the chip. V_{REF} should be well regulated and used only for the A/D converter. The V_{REF} supply can be between 4.5V and 5.5V and needs to be able to source around 5 mA. Figure 15 shows all of these connections.

Note that if only ratiometric information is desired, V_{REF} can be connected to V_{CC}. In addition, if the A/D converter is not being used, V_{REF} must be connected to V_{CC} and ANGND to V_{SS} for Port0 to work as a digital port.

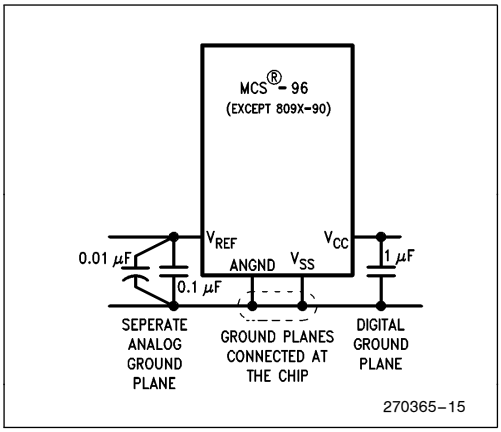


Figure 15. Supply Decoupling

Getting More Resolution

Given that the A/D converter can convert an analog input ranging from 0 volts to 5 volts into 1024 steps of 5 millivolts each, the desire for more resolution can come from three basic needs – need extra LSB, need extra MSB, need BOTH.

The configuration shown in Figure 16 can be used to solve each of the "more resolution" problems. This set-up requires the use of two input channels with different offsets and gains.

When the 5 millivolt step size of the A/D is too large for the application requirements, but the 5 volt range is sufficient, the system needs an "extra LSB". For example, an application requiring 2.5 millivolt steps over a 5 volt range needs an 11-bit conversion result. The 11th bit needs to be added to the least significant side of the 10- bit result (the "right"). This can be achieved using the circuit of Figure 16.

If both channels are set for a gain of 2, with channel 1 offset to 2.5 volts, the 5 volt input range is split into 2.5 volt ranges that are amplified by two before being input to the A/D. While V_{IN} is between 0 and 2.5 volts, channel 0 will be providing a proportional voltage between 0 volts and 5 volts to the A/D converter. Channel 1 will be clamped to 5 volts. When V_{IN} rises above 2.5 volts, channel 1 will begin to output a proportional voltage between 0 volts and 5 volts to the A/D converter and channel 0 will be clamped at 5 volts. Using this method, an 11-bit (2048 step) result is created with 2.5 millivolt steps (i.e. an extra LSB).

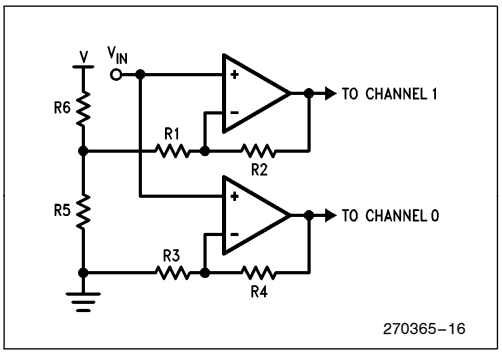


Figure 16. A Flexible Input Circuit

It is useful to note that only one conversion per sample will be required if the software keeps track of which channel is active. The only time that two conversions will be required for one sample is when the voltage crosses the midpoint.

The second reason that "more resolution" is requested is the need for an "extra MSB". When the converter's input voltage range is too small (5 volts when 10 volts is needed), but 5 millivolt steps over the actual input voltage range is sufficient, an extra bit is needed on the most significant ("left") side of the 10-bit result. The circuit of Figure 16 can also be used, with different gains and offsets, to satisfy this extra MSB need by splitting the 10 volt range into 5 volt ranges.

If both channels of Figure 16 are set for unity gain, and channel 1 is offset to 5 volts, an 11-bit conversion result with 5 millivolt steps is available. While V_{IN} is in the lower half of its range (0 volts to 5 volts), channel 0 will be active. While V_{IN} is in the upper half of its range (5 volts to 10 volts), channel 1 will be active. Thus, an extra MSB is created.

For applications requiring multiple extra bits of result, the solutions can become more "elegant" (i.e. elaborate). However, it is profitable to first squeeze the most out of the now familiar circuit in Figure 16.

Assume that the analog input, V_{IN}, ranges from 0 volts to 10 volts, and it is desired to measure this range in 2.5 millivolt steps. This requires two extra bits of result – one extra MSB and one extra LSB. A simple extrapolation of the preceding discussion of creating extra bits might have the designer planning to tieup four channels of the multiplexer needlessly. Needlessly, that is, if the application is a typical control application where the high accuracy requirements are only important in the "normal" operating range of the process. Outside of the normal operating range is the "possible" operating range which must be measured, but with less stringent requirements.

Since the requirements of the normal range set the necessary LSB weight, and the extent of the possible range sets the maximum voltage span, it follows that only two channels need to be used (Figure 16). Channel 0 would be set with a gain that compressed the possible V_{IN} range to 5 volts, while channel 1 would be offset to the normal operating range and would have a gain of two to expand this region of critical interest. With this ap-

proach, 100 percent of the normal operating range is digitized in 2.5 millivolt steps, while 100 percent of the possible range is digitized in 10 millivolt steps.

Unfortunately, not all high resolution applications can be described as a process with a small region of in-control operation, where the process is out-of-control outside of that small region. For example, it is necessary to measure airflow in an engine controlling carburetion. The air flow at idle is likely to be several orders-of-magnitude lower than the airflow at full RPM. The process needs to be in tight control over the entire range, not only when the engine is at half-speed.

When it is desired to measure a process with a fixed percent of error throughout a range spanning several orders-of-magnitude, a non- linear input buffer becomes attractive. For example, assume that the analog signal that needs to be digitized can vary from 1 millivolt to 25 volts and describes a physical process that must be represented digitally with 1 percent error at any point in the possible input range. A linear solution to this application would require a converter with a 10 microvolt LSB (1% $\times$ 1 mV), and a resolution of 22 bits (25 V/10 microvolts). This is clearly undesirable.

The use of a log input buffer to compress the 25 volt range logarithmically to 5 volts would satisfy the application requirements. The input would range from 1 millivolt to 25 volts with the output ranging from 0 volts to 5 volts proportionally to the log of V_{IN}/1mV. Each one-percent change in the input voltage would change the output voltage by 5 millivolts (one count). The antilog could be taken in software using a lookup table, or the control calculations could be performed in a log base.

Simple inexpensive log-amps can be built as in Figure 17, or high- accuracy, self-contained log-amps can be purchased. Which is chosen depends upon the application tradeoffs of price and performance.

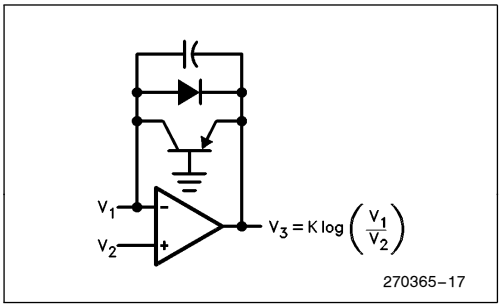


Figure 17. A Low-Cost Log Amplifier

Other techniques become available for consideration in systems that have slow sample rate requirements, but very high resolution requirements. In addition to the methods described above, which require external hardware, software filtering or other post-processing of the conversion results can be productive. Each method relies upon the ability to sample the analog input much faster than the system requires an analog input.

When resolution is limited by filterable noise, perhaps the most straightforward approach to post-processing is to oversample the input by a factor of N and digitally low-pass filter the data (i.e. weighted rolling average). A result would be reported to the rest of the system every N samples (Figure 18). A low-pass filter can increase the signal- to-noise ratio (SNR) by a factor of N (see bibliography). However, care must be taken to be certain that the input voltage varies slowly with respect to the sampling rate.

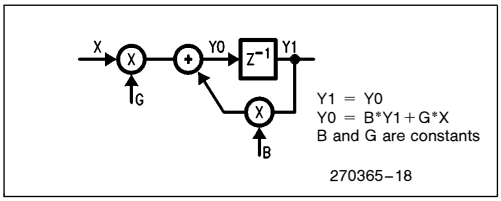


Figure 18. A Low Pass Filter

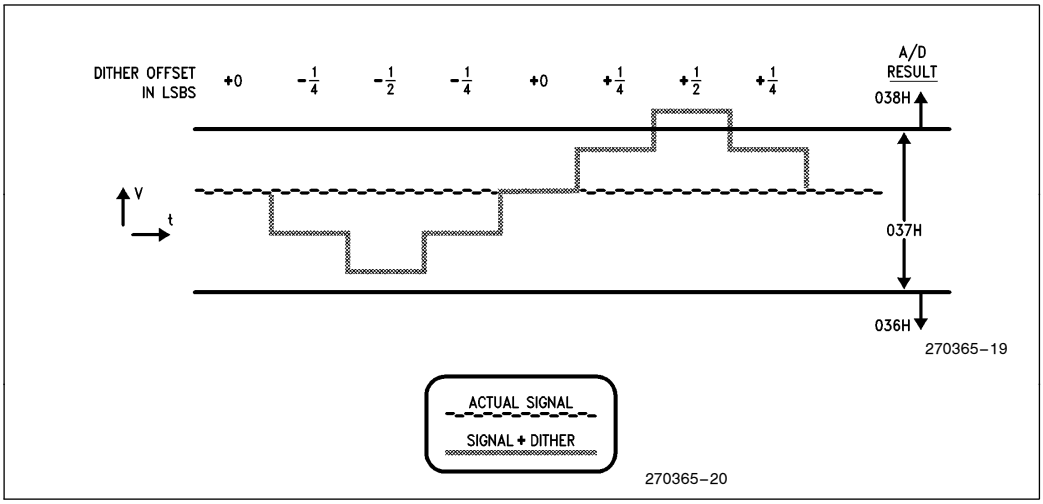


Figure 19. Dither

Another approach to creating more resolution is called "synchronized dither". Figure 19 shows an input voltage that is constant somewhere between two code transition points. This input is "dithered" by adding a small periodic waveform (¼ LSB steps) to the input while performing an A/D conversion synchronized with each dither step. Every time the dither completes a full cycle, the eight conversion results are averaged to form one digitized value. Since the dither is periodic and symmetrical about 0 volts, its average impact on the input voltage is 0 volts.

The creation of extra resolution can be seen with the example shown in Figure 19. Without dither, the input voltage would always convert to 37H. With dither, one-eighth of the conversions would be 38H and ⅞ of the conversions would be 37H. If every eight conversions were averaged, the result would be 37H + ⅛ LSB. The possible results given a four level dither, where the input voltage was always within the 37H code width, would be

36H + ⅝
36H + ⅞
37H + 0
37H + ⅛
37H + ⅜

Hence, four new levels exist (two bits).

Dither will only create more resolution up to the limit of the A/D converter comparator's ability to distinguish voltages. Since MCS-96 converter repeatability error is typically around 1 millivolt to 1.25 millivolts, ¼ LSB dither is the practical limit if no other processing is done. Figure 20 shows a simple method by which the input voltage could be dithered under software control.

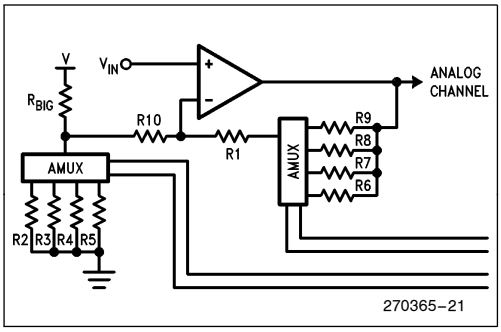


Figure 20. Software Controlled Offset and Gain

While only a few of the more obvious interfacing techniques were described here, there are as many innovative interfacing tricks as there are designers.

CONCLUSION

This application note provides a fundamental understanding of MCS-96 analog acquisition for the digital designer. Since answering the limitless number of analog circuit design questions is beyond the scope of this document, it is expected that analog design manuals and the large body of publicly available applications literature will be consulted for detailed design hints. Furthermore, the applications literature of monolithic analog acquisition system manufacturers should be consulted since the suggestions presented therein are largely transportable to any A/D system.

APPENDIX A
A/D GLOSSARY OF TERMS

Figures 2, 4 and 7 display many of these terms.

ABSOLUTE ERROR—The maximum difference between corresponding actual and ideal code transitions. Absolute Error accounts for all deviations of an actual converter from an ideal converter.

ACTUAL CHARACTERISTIC—The characteristic of an actual converter. The characteristic of a given converter may vary over temperature, supply voltage, and frequency conditions. An Actual Characteristic rarely has ideal first and last transition locations or ideal code widths. It may even vary over multiple conversions under the same conditions.

BREAK-BEFORE-MAKE—The property of a multiplexer which guarantees that a previously selected channel will be deselected before a new channel is selected. (e.g. the multiplexer will not short inputs together.)

CHANNEL-TO-CHANNEL MATCHING—The difference between corresponding code transitions of actual characteristics taken from different channels under the same temperature, voltage and frequency conditions.

CHARACTERISTIC—A graph of input voltage versus the resultant output code for an A/D converter. It describes the transfer function of the A/D converter.

CODE—The digital value output by the converter.

CODE CENTER—The voltage corresponding to the midpoint between two adjacent code transitions.

CODE TRANSITION—The point at which the converter changes from an output code of Q, to a code of Q + 1. The input voltage corresponding to a code transition is defined to be that voltage which is equally likely to produce either of two adjacent codes.

CODE WIDTH—The voltage corresponding to the difference between two adjacent code transitions.

CROSSTALK—See "Off-Isolation".

D.C. INPUT LEAKAGE—D.C. Leakage current of an analog input pin.

DIFFERENTIAL NON-LINEARITY—The difference between the ideal and actual code widths of the terminal based characteristic of a converter.

FEEDTHROUGH—Attenuation of a voltage applied on the selected channel of the A/D converter after the sample window closes.

FULL-SCALE ERROR—The difference between the expected and actual input voltage corresponding to the full-scale code transition.

IDEAL CHARACTERISTIC—A characteristic with its first code transition at V_{IN} = 0.5 LSB, its last code transition at V_{IN} = (V_{REF} − 1.5 LSB) and all code widths equal to one LSB.

INPUT RESISTANCE—The effective series resistance from the analog input pin to the sample capacitor.

LSB - LEAST SIGNIFICANT BIT—The voltage value corresponding to the full-scale voltage divided by 2n, where n is the number of bits of resolution of the converter. For a 10-bit converter with a reference voltage of 5.12 volts, one LSB is 5.0 mV. Note that this is different than digital LSBs, since an uncertainty of two LSBs, when referring to an A/D converter, equals 10 mV. (This has been confused with an uncertainty of two digital bits, which would mean four counts, or 20 mV.)

MONOTONIC—The property of successive approximation converters which guarantees that increasing input voltages produce adjacent codes of increasing value, and that decreasing input voltages produce adjacent codes of decreasing value.

NO MISSED CODES—For each and every output code, there exists a unique input voltage range which produces that code only.

NON-LINEARITY—The maximum deviation of code transitions of the terminal based characteristic from the corresponding code transitions of the actual characteristic of a converter.

OFF-ISOLATION—Attenuation of a voltage applied on a deselected channel of the A/D converter. (Also referred to as Crosstalk.)

REPEATABILITY—The difference between corresponding code transitions from different actual characteristics taken from the same converter on the same channel at the same temperature, voltage and frequency conditions.

RESOLUTION—The number of input voltage levels that the converter can unambiguously distinguish between. Also defines the number of useful bits of information which the converter can return.

SAMPLE DELAY—The delay from receiving the start conversion signal to when the sample window opens.

SAMPLE DELAY UNCERTAINTY—The variation in the Sample Delay.

SAMPLE TIME—The time that the sample window is open.

SAMPLE TIME UNCERTAINTY—The variation in the sample time.

SAMPLE WINDOW—Begins when the sample capacitor is attached to a selected channel and ends when the sample capacitor is disconnected from the selected channel.

SUCCESSIVE APPROXIMATION—An A/D conversion method which uses a binary search to arrive at the best digital representation of an analog input.

TEMPERATURE COEFFICIENTS—Change in the stated variable per degree centigrade temperature change. Temperature coefficients are added to the typical values of a specification to see the effect of temperature drift.

TERMINAL BASED CHARACTERISTIC—An Actual Characteristic which has been rotated and translated to remove zero offset and full-scale error.

V_{CC} REJECTION—Attenuation of noise on the V_{CC} line to the A/D converter.

ZERO OFFSET—The difference between the expected and actual input voltage corresponding to the first code transition.

APPENDIX B
CAPACITIVE INTERPOLATION

A successive approximation A/D converter needs an internal D/A converter of the same resolution as the desired A/D result. A 10-bit D/A could have been made using a string of 1024 resistors connected from the analog reference at one end to ground at the other end. Although this would be technically ideal, such a circuit would be enormous. Therefore, a method was developed to generate the needed reference voltages using a small area of silicon so that an on-chip 10-bit A/D converter would be economical.

The method used relies upon a 256-resistor chain to generate reference voltages in 20mV (5.12V/256) steps while two ratioed capacitors are used to capacitively "interpolate" voltages in-between the resistor tap voltages. The area of the 256-resistor chain together with the capacitors is one-fourth the area of the would-be 1024 resistor chain.

Before beginning a detailed description of the capacitive part of the conversion process, it is necessary to understand a few details about the resistor chain.

There are 256 resistors connected in series from the analog reference to analog ground. The actual value of the resistors only impacts the current through the reference pin. If every resistor in the chain is the same value the converter will function properly.

To reduce resistor-to-resistor variation, the chain is folded in half, and then in an accordion fashion to produce a 16 × 16 block of resistors. This minimizes the sensitivity of the array to processing gradients, while also allowing the array to be addressed roughly similar to a 16 × 16 memory array.

As explained earlier, it is desired for the A/D converter to have its first code transition at ½ LSB followed by subsequent code widths 1 LSB wide.

To accomplish this, each resistor is tapped in its center rather than between resistors. For example, the first resistor tap is half-way up the first resistor. This means that the zero resistor tap will output 10mV (20mV/2). When calculating the voltage on a certain resistor tap, you must add 10mV to the product of the tap number and 20mV.

The internal connections while an analog input is being sampled are shown in Figure B1a. Once sampling is complete, the analog input is disconnected and the comparator inputs are no longer clamped to V_{BIAS} (Figure B1b).

During the sample window (Figure B1a), V_{ANIN} and V_{OFS} control the amount of charge stored in C_A and C_B (V_{OFS} controls the converter offset). Once the sample window closes (Figure B1b), voltages applied to V_{IN} and V_{IN2} will add or subtract charge proportional to ($V_{ANIN} - V_{IN}$) on C_A and ($V_{OFS} - V_{IN2}$) on C_B. Unless a voltage is applied to V_{IN} and V_{IN2}. The inverting comparator input of Figure B1b will remain at V_{BIAS} due to the charges on C_A and C_B. The non-inverting comparator input will always remain at V_{BIAS} and serves as a reference.

If a V_{IN}, V_{IN2} combination is applied which causes the non-inverting input to drop below V_{BIAS} the comparator will output to a 1 to indicate that the applied voltage was lower than the original V_{ANIN}. To better understand how the circuit works, Figure B2 shows the superposition analysis used to form the equation for V_{OUT}, given initial charge on C_A and C_B and new input voltages V_{IN} and V_{IN2}.

Adding the independent effects shown in Figure B2 we have:

$$V_{OUT} = V1 + V2 + V3 + V4$$

$$V_{OUT} = V_{IN}\left(\frac{C_A}{C_A+C_B}\right) + V_{IN2}\left(\frac{C_B}{C_A+C_B}\right) + V_{AI}\left(\frac{C_A}{C_A+C_B}\right) + V_{BI}\left(\frac{C_B}{C_A+C_B}\right)$$

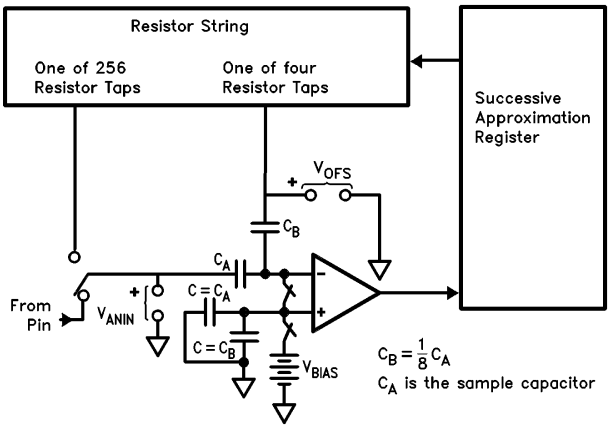
$$V_{OUT} = (V_{IN} + V_{AI})\frac{C_A}{C_A + C_B} + (V_{IN2} + V_{BI})\frac{C_B}{C_A + C_B} \quad \text{(I)}$$

The initial conditions on C_A and C_B are set-up as shown in Figure B3.

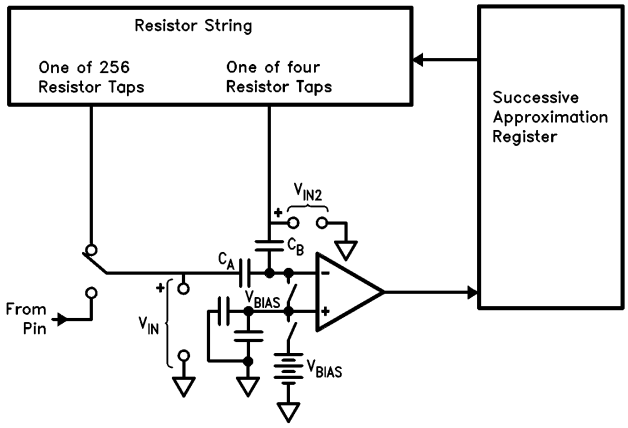
We can see that:

$$V_{AI} = V_{BIAS} - V_{ANIN} \quad \text{(II)}$$

$$V_{BI} = V_{BIAS} - V_{OFS} \quad \text{(III)}$$



(a). Connections during the sample window



(b). Connections after the sample window closes

Figure B1

Substituting II and III into I we get:

$$V_{OUT} = (V_{IN} + V_{BIAS} - V_{ANIN}) \frac{C_A}{C_A + C_B} +$$

$$(V_{IN2} + V_{BIAS} - V_{OFS}) \frac{C_B}{C_A + C_B} \qquad \text{(IV)}$$

V_{OUT} becomes the input voltage to the comparator which ideally presents no load. The only way to make V_{OUT} approach the value of V_{BIAS} (after V_{BIAS} is removed) is to apply a voltage combination which makes equation IV evaluate to V_{BIAS}. If we had an infinitely variable internal voltage reference to use, we could just set the reference on V_{IN} to the value of V_{ANIN} and make $V_{INZ} = V_{OFS}$.

We would then have, from IV:

$$V_{IN} = V_{ANIN}, V_{IN2} = V_{OFS}$$

However, using a 256-resistor chain to provide references, we can find a V_{IN}, V_{INZ} combination which can bring V_{OUT} close to the value of V_{BIAS}. The 256-resistor chain provides a reference voltage in 20 mV steps. We can then take separate taps of the resistor chain and connect them to V_{IN} and V_{IN2}. The voltage attached to V_{IN} will couple to V_{OUT} by a factor of $C_A/(C_A + C_B) = 8/9$ from EQN IV. The voltage attached to V_{IN2} will couple to V_{OUT} by a factor of $C_B/(C_A + C_B)$. The ratio of the impacts on V_{OUT} of V_{IN} versus V_{IN2} is:

$$\left(\frac{\partial V_{OUT}}{\partial V_{IN}}\right) \div \left(\frac{\partial V_{OUT}}{\partial V_{IN2}}\right) = (8/9)/(1/9) = 8$$

Therefore, a voltage change on V_{IN} will affect the voltage seen at V_{OUT} eight times more than the same change placed on V_{IN2}.

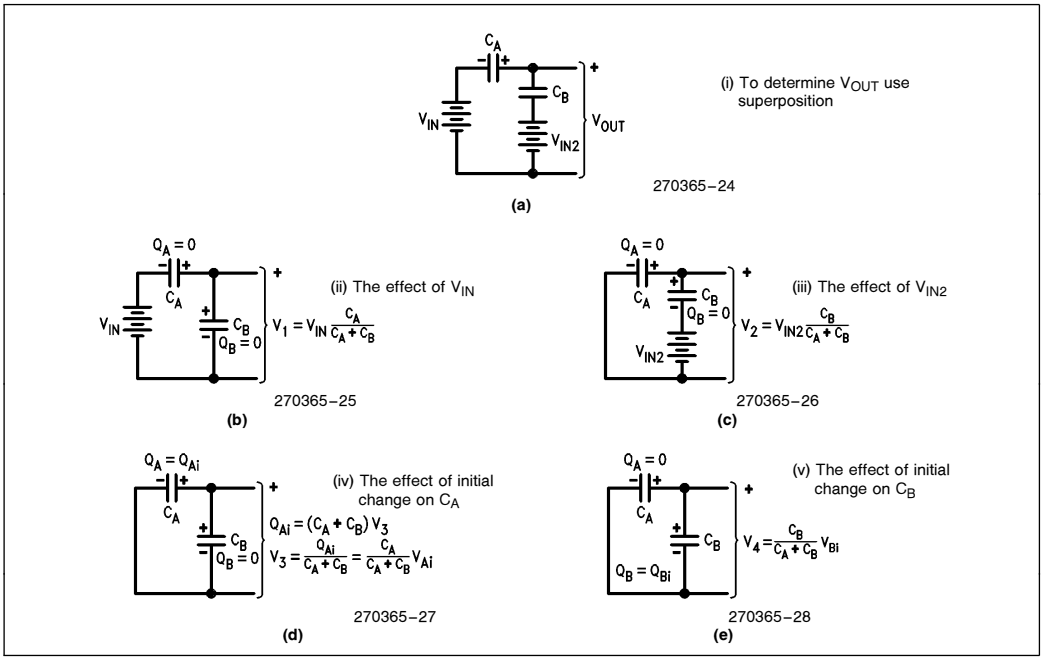


Figure B2. Superposition Analysis of comparator input voltage

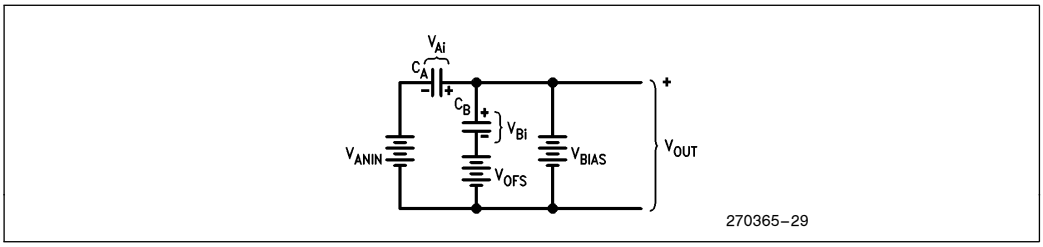


Figure B3. Initial Conditions

For example, assume the actual input voltage V_{ANIN} was 2.50mV during the sample window. Using EQN IV, and assuming $V_{BIAS} = 3V$ and $V_{OFS} = 70mV$, we substitute and find:

$$V_{OUT} = (V_{IN} + 2.9975) \times (8/9) + (V_{IN2} + 2.93) \times (1/9) \quad (V)$$

Using successive approximation, the first trial input voltage attempted corresponds to the digital code 0111 1111 11b (127 × 20mV + 10mV). This means that the voltage applied to V_{IN} will be the 0111 1111b tap and the voltage applied to V_{IN2} will be the 0110b tap (6 × 20mV + 10mV = 3 LSB). Substituting these values into EQN V we have:

$$V_{OUT} = (2.550 + 2.9975) \times (8/9) + (0.130 + 2.93) \times (1/9) \quad (V)$$

$$V_{OUT} = 4.931 + 0.34 = 5.271$$

Since the 3V reference is lower than V_{OUT} with these inputs, the comparator will output a 0 which is placed in the MSB of the successive approximation register. The next most significant bit of the SAR is then zero'd and the new ladder tap applied to V_{IN}. The result of this second comparison, and the subsequent comparisons are shown in Table B1. The C program used to generate Table B1 is listed in Listing B1.

The value selected for V_{OFS} during the sample window may not be obvious. The purpose of V_{OFS} is to inject a constant offset in the sampling process so that the converter's first code transition will occur at 2.5mV.

Using EQN IV we can quickly see why V_{OFS} is chosen to be the fourth resistor tap (4 × 20mV + 10mV = 70mV). For $V_{ANIN} = 2.5mV$, we want V_{OUT} to evaluate to V_{BIAS} when the SAR is OH.

$$V_{OUT} = \{(0.20\ mV + 10\ mV) + (V_{BIAS} - 2.5\ mV)\} \times (8/9) + \{(0.20\ mV + 10mV) + (V_{BIAS} - 70\ mV)\} \times (1/9)$$

$$V_{OUT} - V_{BIAS} = 7.5\ mV \times (8/9) - 60\ mV \times (1/9) = 0$$

Therefore, if $V_{OFS} = 70$ mV, the converter's first code transition will be when $V_{ANIN} = 2.5$ mV.

Table B1. Conversion Simulation

```
A to D simulator. (center taps) . . With
VIN = 0.002500
VCENT = 3.000000 VOFF = 0.070000
SAR = 1FFH ( 511)   VOUT = 5.271111
SAR =  FFH ( 255)   VOUT = 4.133333
SAR =  7FH ( 127)   VOUT = 3.564444
SAR =  3FH (  63)   VOUT = 3.280000
SAR =  1FH (  31)   VOUT = 3.137778
SAR =   FH (  15)   VOUT = 3.066667
SAR =   7H (   7)   VOUT = 3.031111
SAR =   3H (   3)   VOUT = 3.013333
SAR =   1H (   1)   VOUT = 3.004444
SAR =   0H (   0)   VOUT = 3.000000
SAR =   1H (   1)    which means 0.005000 volts
```

```
#include "CTYPE.H"
#include "STDIO.H"
/* example invocation lines

  a2dsim 0.0025  3.0    0.07          p
         Vin     Vbias   Vofs    print to screen and lp

  a2dsim 0.0075  3.0    0.07
         Vin     Vbias   Vofs    print to screen only
*/


int main(k, argv)
int k;
char *argv[];
        {                                       /* main */
        FILE *fp, *fopen();
        double initial_conditions, vin, vout, vcent, voff, v89, v19;
        unsigned int sar = 0x3FF;
        unsigned int mask = 0x200;
        unsigned int count = 0;
        unsigned int printon;
        if (strcmp(argv[0], "run") == 0)
                count++;
        if ((k != (4 + count)) & (k != (5 + count)))
                {
                printf("\nInvocation error!\n");
                return;
                }
        count++;
        sscanf(argv[count++], "%lf", &vin);
        sscanf(argv[count++], "%lf", &vcent);
        sscanf(argv[count++], "%lf", &voff);
        if (count == k)
                printon = 0;
        else printon = 1;

        printf("A to D simulator.(center taps)..");

        if (printon)
                {
                if ((fp = fopen("\prn:", "w")) == 0)
                        {
                        printf("\nCan't open printer\n");
                        return;
                        }
                }
        if (printon)
                fprintf(fp, "A to D simulator..");

        printf(" with \nVin = %f\nVcent = %f\nVoff = %f\n", vin, vcent, voff);
        if (printon)
                fprintf(fp, " with \nVin = %f\nVcent = %f\nVoff = %f\n",
                        vin, vcent, voff);

        initial_conditions = ((8.0 / 9.0) * (vcent - vin))
             + ((1.0 / 9.0) * (vcent - voff));
        v89 = 8.0 / 9.0;
        v19 = 1.0 / 9.0;
```

270365–A5

Listing B1. A/D Converter Simulator

```
sar ^= mask;
printf("SAR = %3xH (%4d)\t", sar, sar);
if (printon)
        fprintf(fp, "SAR = %3xH (%4d)\t", sar, sar);
for (count = 0;  count < 10;  count++)
        {

        vout = (v89 * (((double) (sar >> 2)) * 0.02 + 0.01))
             + (v19 * (((double) ((sar & 3) << 1)) * 0.02 + 0.01))
             + initial_conditions;
        if (vout < vcent)
              sar |= mask;
        mask >>= 1;
        sar ^= mask;
        printf("Vout = %f\nSAR = %3xH (%4d)\t", vout, sar, sar);
        if (printon)
                fprintf(fp, "Vout = %f\nSAR = %3xH (%4d)\t",
                        vout, sar, sar);
        }
printf(" which means %f volts\n\n", (double) sar * 0.005);
if (printon)
        fprintf(fp, " which means %f volts\n\n", (double) sar * 0.005);
return;
}                                       /* main */
```

270365–A6

Listing B1. A/D Converter Simulator (Continued)

APPENDIX C
ERROR FORMULAS

The following C program listing contains the routines used to calculate A/D performance in the Embedded Controller Applications lab. Most of the routines require floating point arrays to operate upon. In the listings, the array x[ ] contains the input voltages corresponding to each code transition of the converter. The array dx[ ] contains the width of the region in which each code transition of the converter could occur. For example, an input voltage of 0.003V may cause code 0 and code 1 to be equally likely outputs. x[0] would then contain 0.0030000. However, 0-to-1 code transitions might be observed infrequently through a range of input voltages from 0.0025V to 0.0035V. dx[0] would then contain 0.0010000 to indicate that there is a 1 millivolt window in which either code could occur. x[ ] and dx[ ] are generated by hardware doing repeated conversions using precision voltage standards to provide the input voltages. The array dd[ ] is used throughout as temporary storage.

Generally, typical data is drawn from x[ ] only. When minimum and maximum data is desired, x[ ] and dx[ ] are used to find the range of possible input voltages that could cause each code. For example, typical zero offset is found by simply subtracting 0.5 LSB from the value of x[0]. But, the minimum and maximum zero offset would be calculated as x[0] - 0.5 LSB $\pm$ dx[0]/2.

The listings are provided to show exactly how performance data is calculated. They are not meant to be compiled by the reader. In fact, they are too incomplete to compile correctly, as some support routines and global data structures are not provided.

```
#include "\DPR\ADTMAC.H"
#include "\DPR\TDBASE.H"
#include "\DPR\RDBASE.H"
#define LSB (nov.avcc/(pow(2.nbits)))
#define FCT (int)(pow(2,nbits) - 2)
#undef min
#undef max
#undef abs

double pow(a, b)
int a, b;
        {                                       /* pow */
        double temp;
        int i;
        temp = 1.0;
        for (i = 1;  i <= ((int) b);  i++,  temp = temp * a)
                ;
        return (temp);
        }                                       /* pow */

double fabs(a)
double a;
        {
        if (a < 0)
                return (-a);
        else return (a);
        }

int min(a, b)
double a, b;
        {
        if (a < b)
                return (1);
        else if (a > b)
                return (2);
        else return (0);
        }

int max(a, b)
double a, b;
        {
        return (min(b, a));
        }

double typzoff(x, dx)
float x[], dx[];
        {
        double pow();
        return (x[0] - (0.5 * LSB));
        }

double maxzoff(x, dx)
float x[], dx[];
        {
        double pow();
        return (x[0] + (dx[0] / 2.0) - 0.5 * LSB);
        }

double minzoff(x, dx)
```

270365-A7

Listing C1. Error Formulas

```
float x[], dx[];
        {
        double pow();
        return (x[0] - (dx[0] / 2.0) - 0.5 * LSB);
        }

double typfse(x, dx)
float x[], dx[];
        {
        double pow();
        return (x[FCT] - (nom.avcc - (1.5 * LSB)));
        }

double minfse(x, dx)
float x[], dx[];
        {
        double pow();
        return ((x[FCT] - (dx[FCT] / 2.0)) - (nom.avcc - (1.5 * LSB)));
        }

double maxfse(x, dx)
float x[], dx[];
        {
        double pow();
        return ((x[FCT] + (dx[FCT] / 2.0)) - (nom.avcc - (1.5 * LSB)));
        }

int xabserror(x, dx, dd, start, stop)   /* transition absolute error */
float x[], dx[], dd[];
unsigned int start, stop;
        {
        double pow(), fabs();
        int i, worst;
        for (i = worst = start;  i <= stop;  i++)
                {
                dd[i] = x[i] - ((double) i + 0.5) * LSB;
                if (fabs(dd[i]) > fabs(dd[worst]))
                        worst = i;
                }
        return (worst);
        }

int xabserrordx(x, dx, dd, start, stop) /* transition absolute error w/dx */
float x[], dx[], dd[];
unsigned int start, stop;
        {
        double pow(), fabs();
        int i, worst;
        double t1, t2;
        for (i = worst = start;  i <= stop;  i++)
                {
                t1 = (x[i] - (dx[i] / 2.0)) - (((double) i + 0.5) * LSB);
                t2 = (x[i] + (dx[i] / 2.0)) - (((double) i + 0.5) * LSB);
                if (fabs(t1) > fabs(t2))
                        dd[i] = t1;
                else dd[i] = t2;
                if (fabs(dd[i]) > fabs(dd[worst]))
                        worst = i;
                }
        return (worst);
        }
```

270365–A8

Listing C1. Error Formulas (Continued)

```
int tbnonlin(x, dx, dd, start, stop)    /* tb nonlin using x only */
float x[], dx[], dd[];
unsigned int start, stop;
        {
        int i, worst;
        double pow(), typzoff(), typfse(), fabs();
        double oadj, qadj;

        oadj = typzoff(x, dx);
        qadj = 1.0 + ((typfse(x, dx) - oadj) / x[stop]);

        for (i = worst = start;  i <= stop;  i++)
                {
                dd[i] = (x[i] - oadj) * qadj - (((double) i + 0.5) * LSB);
                if (fabs(dd[i]) > fabs(dd[worst]))
                        worst = i;
                }
        return (worst);
        }

int tbnonlindx(x, dx, dd, start, stop)  /* tb nonlin using x and dx */
float x[], dx[], dd[];
unsigned int start, stop;
        {
        int i, worst;
        double pow(), typzoff(), typfse(), fabs();
        double oadj, qadj, t1, t2;

        oadj = typzoff(x, dx);
        qadj = 1.0 + ((typfse(x, dx) - oadj) / x[stop]);
```

270365–A9

Listing C1. Error Formulas (Continued)

```
        for (i = worst = start;  i <= stop;  i++)
                {
                t1 = (x[i] - (dx[i] / 2.0) - oadj) * gadj - (((double) i + 0.5) * LSB);
                t2 = (x[i] + (dx[i] / 2.0) - oadj) * gadj - (((double) i + 0.5) * LSB);
                if (fabs(t1) > fabs(t2))
                        dd[i] = t1;
                else dd[i] = t2;
                if (fabs(dd[i]) > fabs(dd[worst]))
                        worst = i;
                }
        return (worst);
        }

int xdnl(x, dx, dd, start, stop)        /* using x only */
float x[], dx[], dd[];
int start, stop;
        {
        int i, worst;
        double pow(), fabs();
        double oadj, gadj;
        double typfse(), typzoff();

        oadj = typzoff(x, dx);
        gadj = 1.0 + ((typfse(x, dx) - oadj) / x[stop]);

        worst = start;
        if (start == 0)
                {
                dd[0] = 0.0;
                start++;
                }
        for (i = start;  i <= stop;  i++)
                {
                dd[i] = (x[i] - oadj) * gadj
                        - (x[i - 1] - oadj) * gadj
                        - LSB;
                if (fabs(dd[i]) > fabs(dd[worst]))
                        worst = i;
                }
        return (worst);
        }

int xdnldx(x, dx, dd, start, stop)      /* using x and dx */
float x[], dx[], dd[];
int start, stop;
        {
        int i, worst;
        double pow(), fabs();
        double t1, t2;
        double oadj, gadj;
        double typfse(), typzoff();

        oadj = typzoff(x, dx);
        gadj = 1.0 + ((typfse(x, dx) - oadj) / x[stop]);


        worst = start;
        if (start == 0)
                {
                dd[0] = dx[0] / 2.0;
```

270365-B0

Listing C1. Error Formulas (Continued)

```
                start++;
                }
        for (i = start;  i <= stop;  i++)
                {
                t1 = (x[i] - (dx[i] / 2.0) - oadj) * gadj
                -(x[i - 1] + (dx[i - 1] / 2.0) - oadj) * gadj
                - LSB;
                t2 = (x[i] + (dx[i] / 2.0) - oadj) * gadj
                -(x[i - 1] - (dx[i - 1] / 2.0) - oadj) * gadj
                - LSB;
                if (fabs(t1) > fabs(t2))
                        dd[i] = t1;
                else dd[i] = t2;
                if (fabs(dd[i]) > fabs(dd[worst]))
                        worst = i;
                }
        return (worst);
        }

int reslevels(x, dx)    /* finds resolution in levels */
float x[], dx[];
        {
        int i, levels, n;
        double pow();

        levels = 1;
        n = (int) pow(2, nbits) - 1;
        if ((x[0] - (dx[0] / 2.0) > 0.0))
                levels++;

        for (i = 1;  i < n;  i++)
                if ((x[i - 1] + (dx[i - 1] / 2.0))
                        < (x[i] - (dx[i] / 2.0) - tparms.fine_step))
                        levels++;
        return (levels);
        }
```

270365-B1

Listing C1. Error Formulas (Continued)

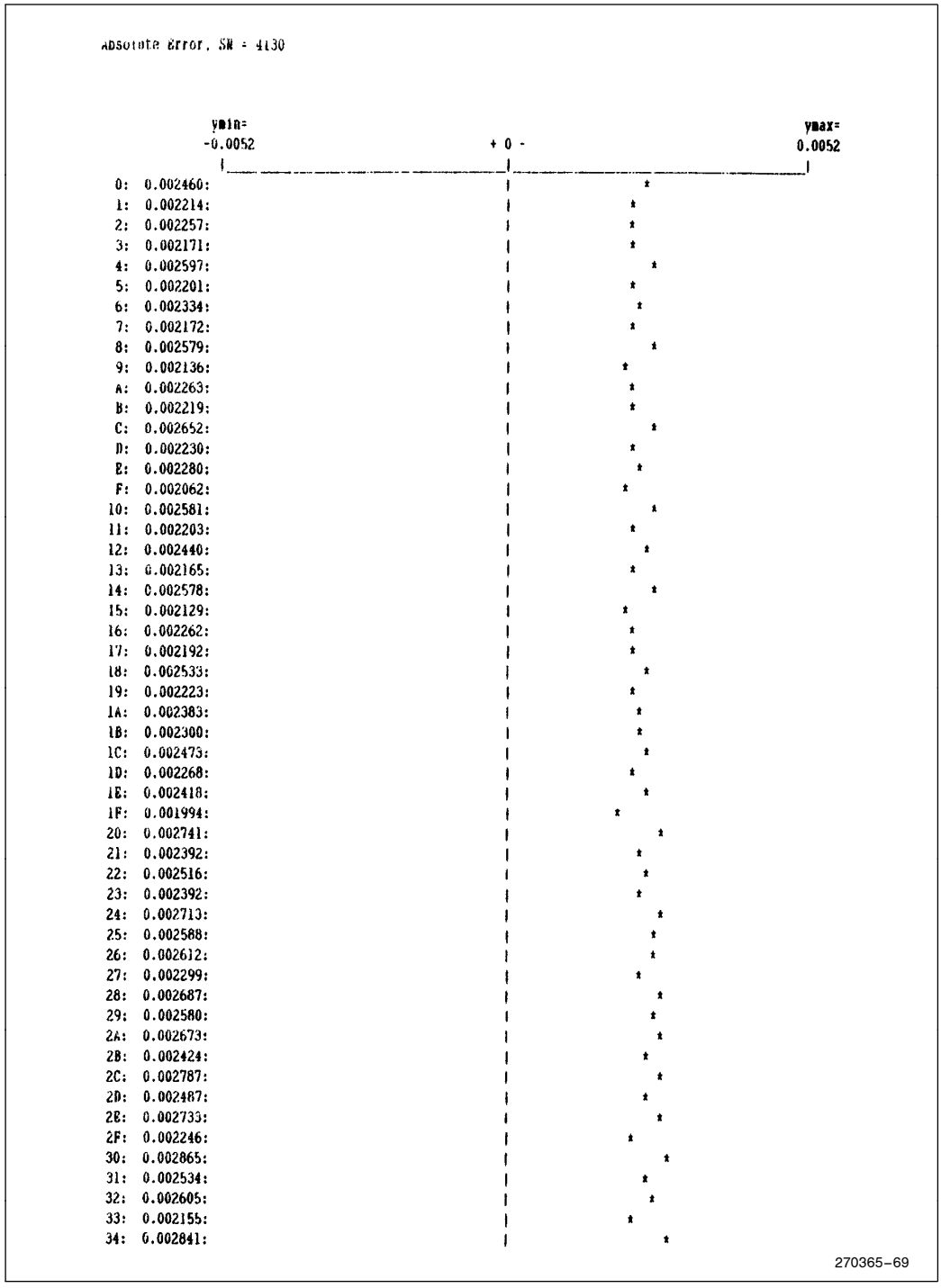
APPENDIX D
SAMPLE CONVERTER DATA

The following pages include printouts describing the performance of an 8097BH. The data shown is for one device and is provided for illustrative purposes only. Users should only rely upon data sheet specifications for the exact device they are designing with.

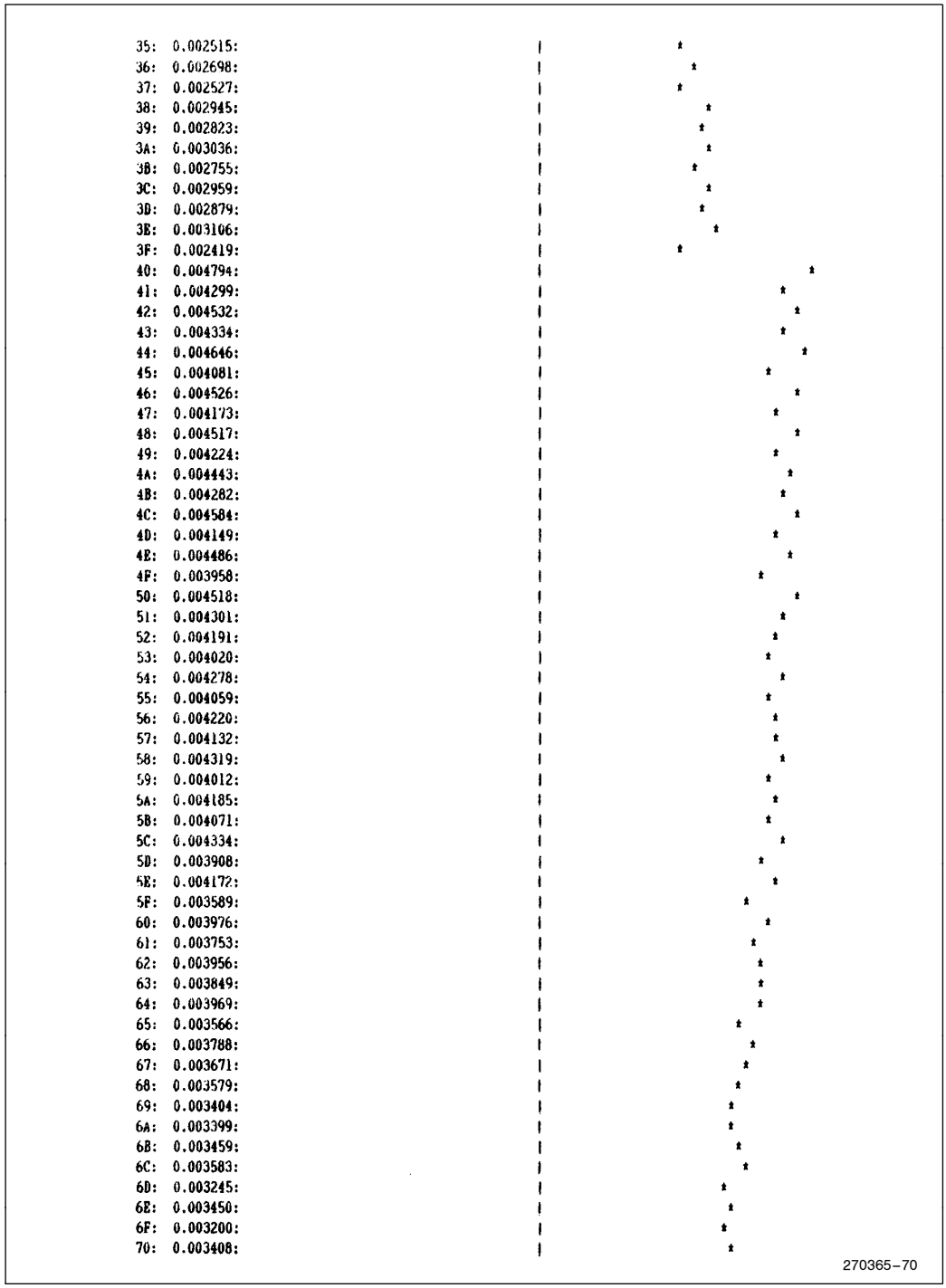
Table D1 summarizes many performance measures for one converter at 25 C, 12 MHz, V_{CC} = 5.00 volts and V_{REF} = 5.120 volts. Following Table D2 are several error plots that describe Absolute Error, Terminal-based Non-Linearity, Differential Non-Linearity and Repeatability for the test device code-by-code. The y-axis in the plots is the error in volts for each code transition, where code transitions make up the x-axis.

Table D1. Sample Converter Data

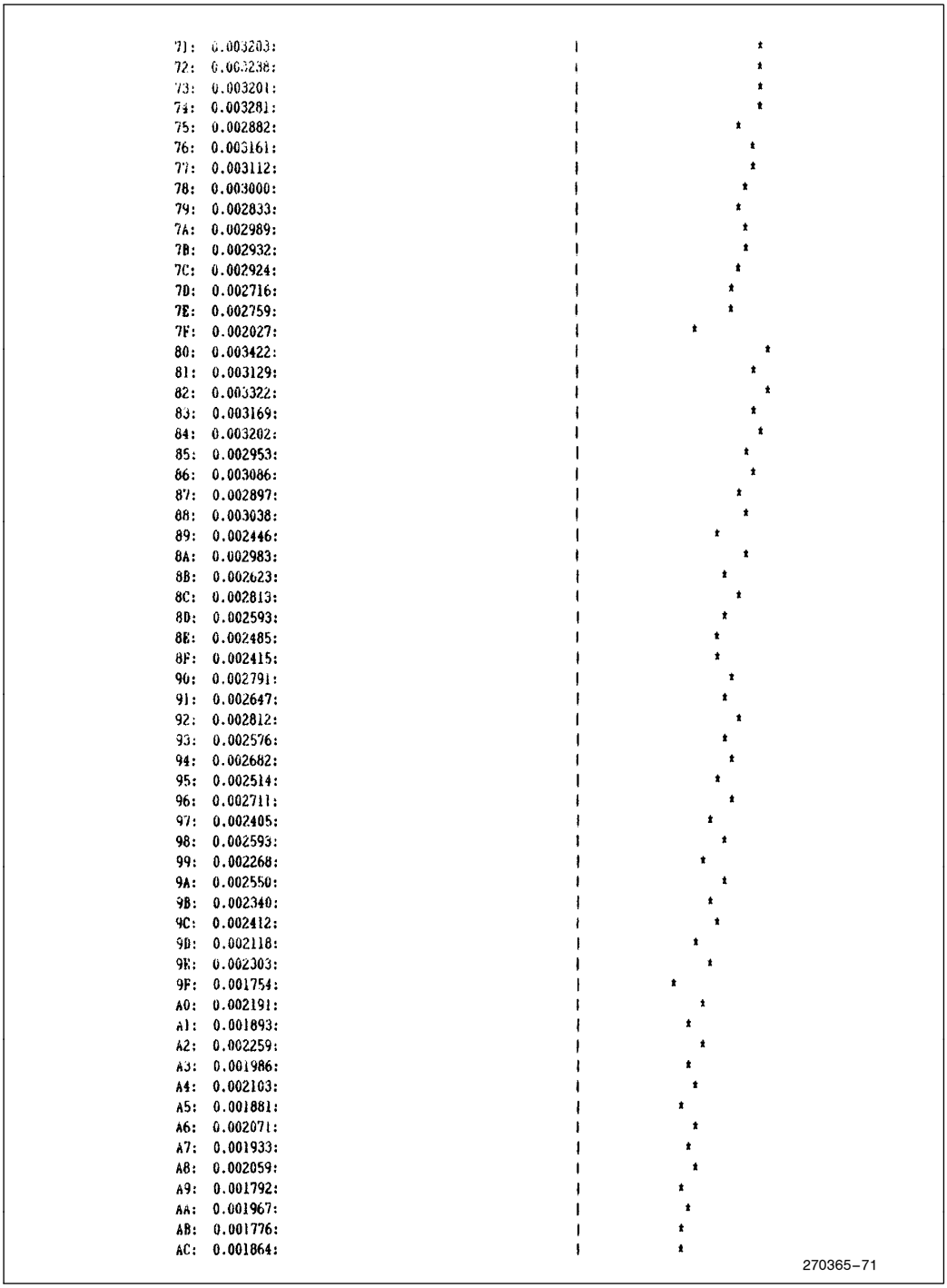
```
Test ID = D0H
sN: 4l30 (l022H)
T = 25.000000
VCC = 5.000000, AVCC = 5.120000
Freq = l2.000000
Chan. = 3
States = l88   Mode = 0H
X0.15 l/28/87
Transition Characterization Parameter Listing
Large Step = 0.00l000 V
Small Step = 0.000l00 V
Endpoints when (l/l00) are wrong

Center is 50 percent

Typical Offset Error = −0.001923
Maximum Offset Error = −0.002460
Maximum Offset Error = −0.001385

Typical FS Error = −0.000566
Maximum FS Error = −0.001254
Minimum FS Error = −0.000120

Absolute Error (typ) 40  = 0.004157
Absolute Error (max) 40  = 0.004795
Absolute Error (min) 325 = 0.001lll

Diff. Non. Lin. Error (max) 40 = 0.003747
Diff. Non. Lin. Error (min) FF = −0.001071

Term. Non. Lin. Error (max) 325 = −0.004102
Term. Non. Lin. Error (min) 40 = 0.002148

Maximum Reliability Error 3Dl = 0.001875
Minimum Reliability Error 3A7 = 0.000974

Resolution is l024 levels.
```



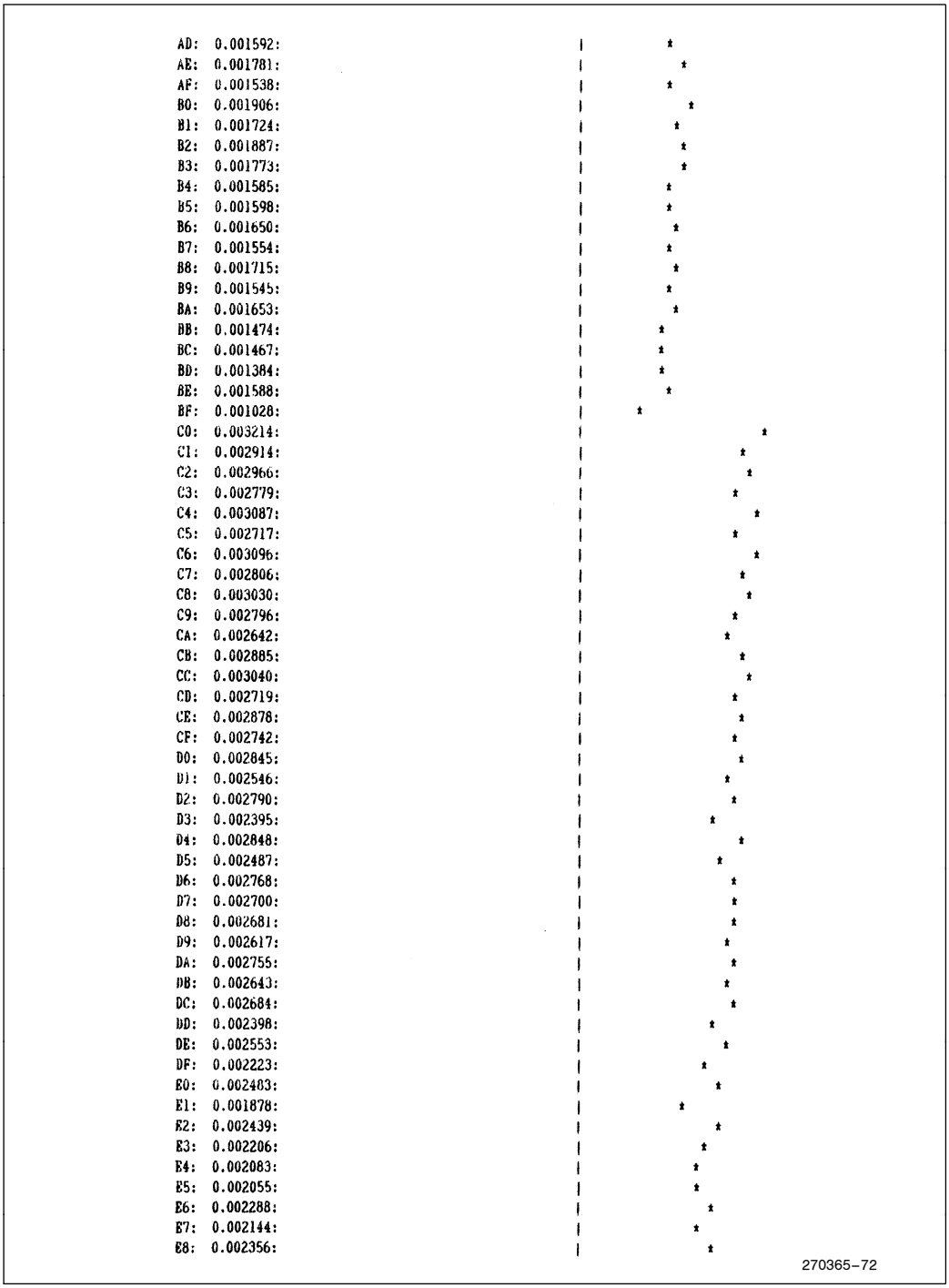
Absolute Error, SN = 4130



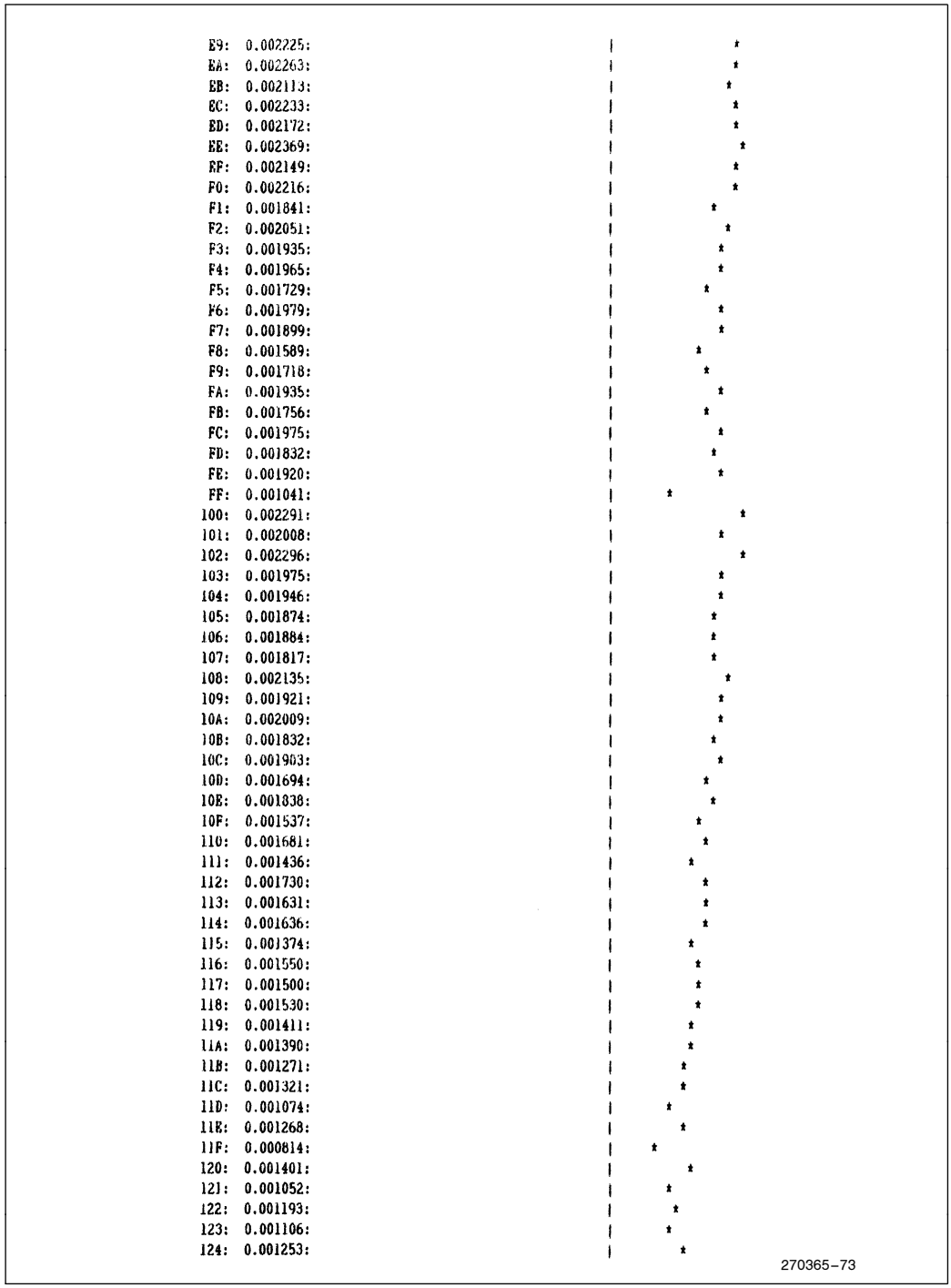
Absolute Error, SN = 4130 (Continued)



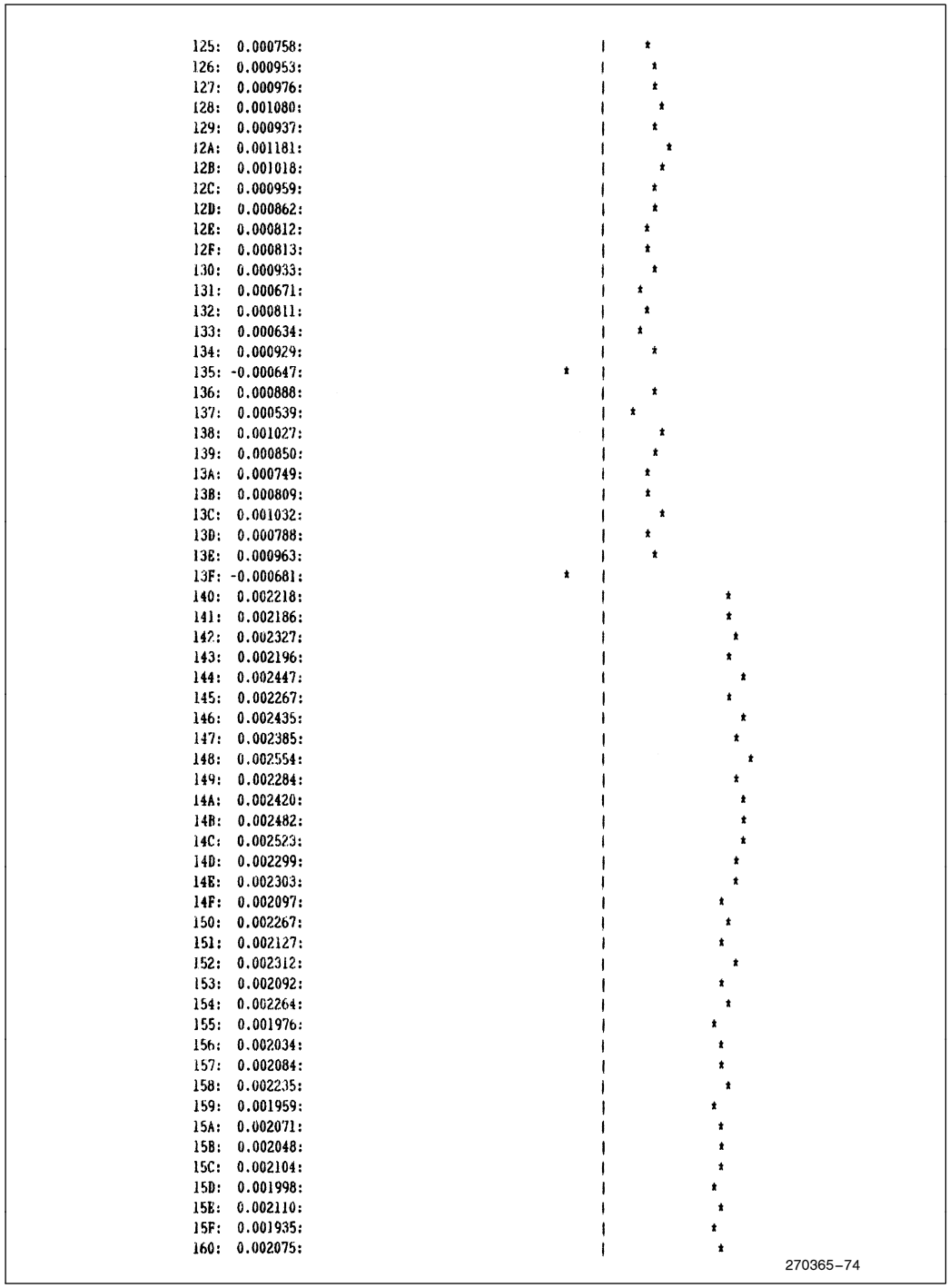
Absolute Error, SN = 4130 (Continued)



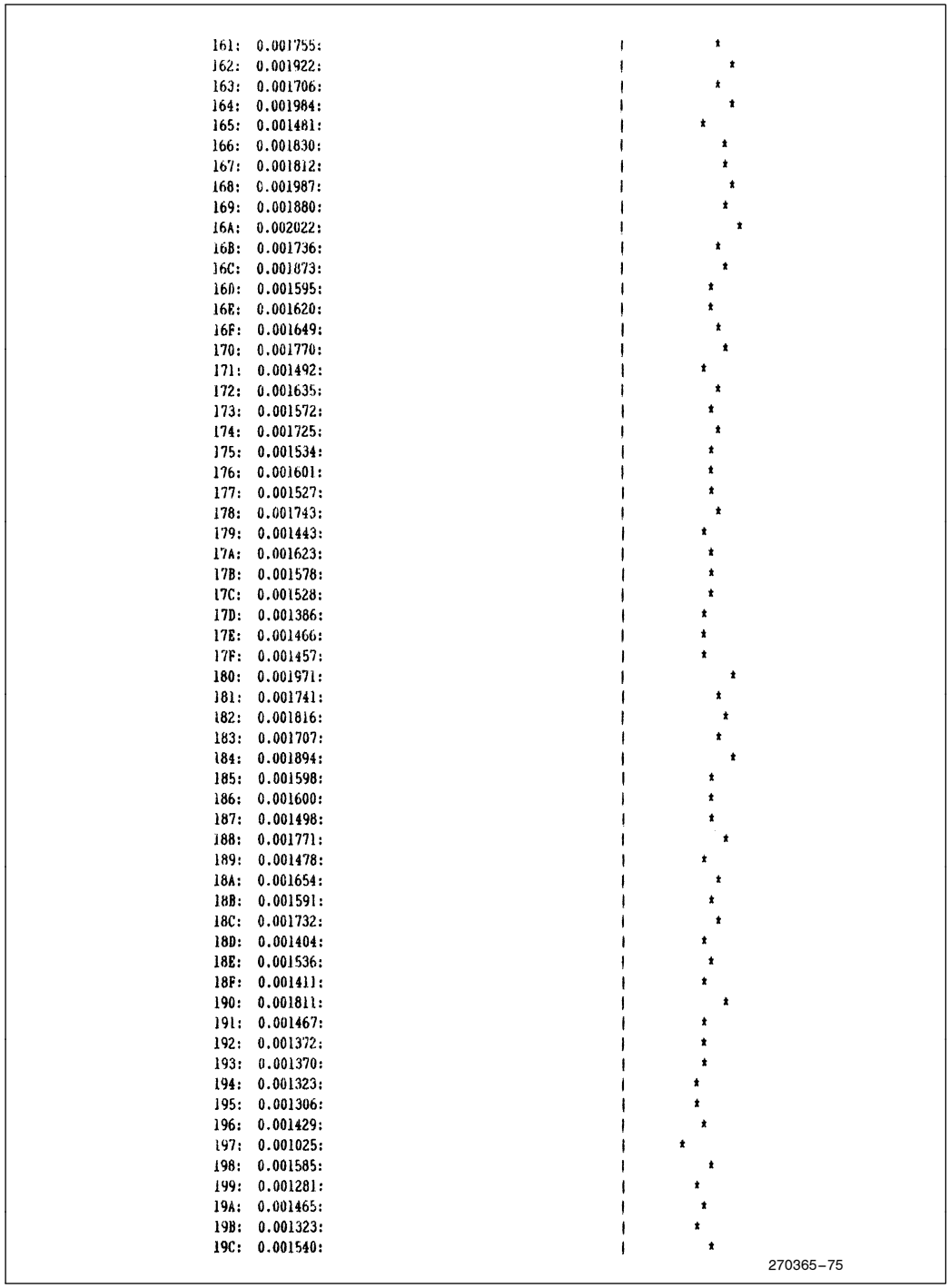
Absolute Error, SN = 4130 (Continued)



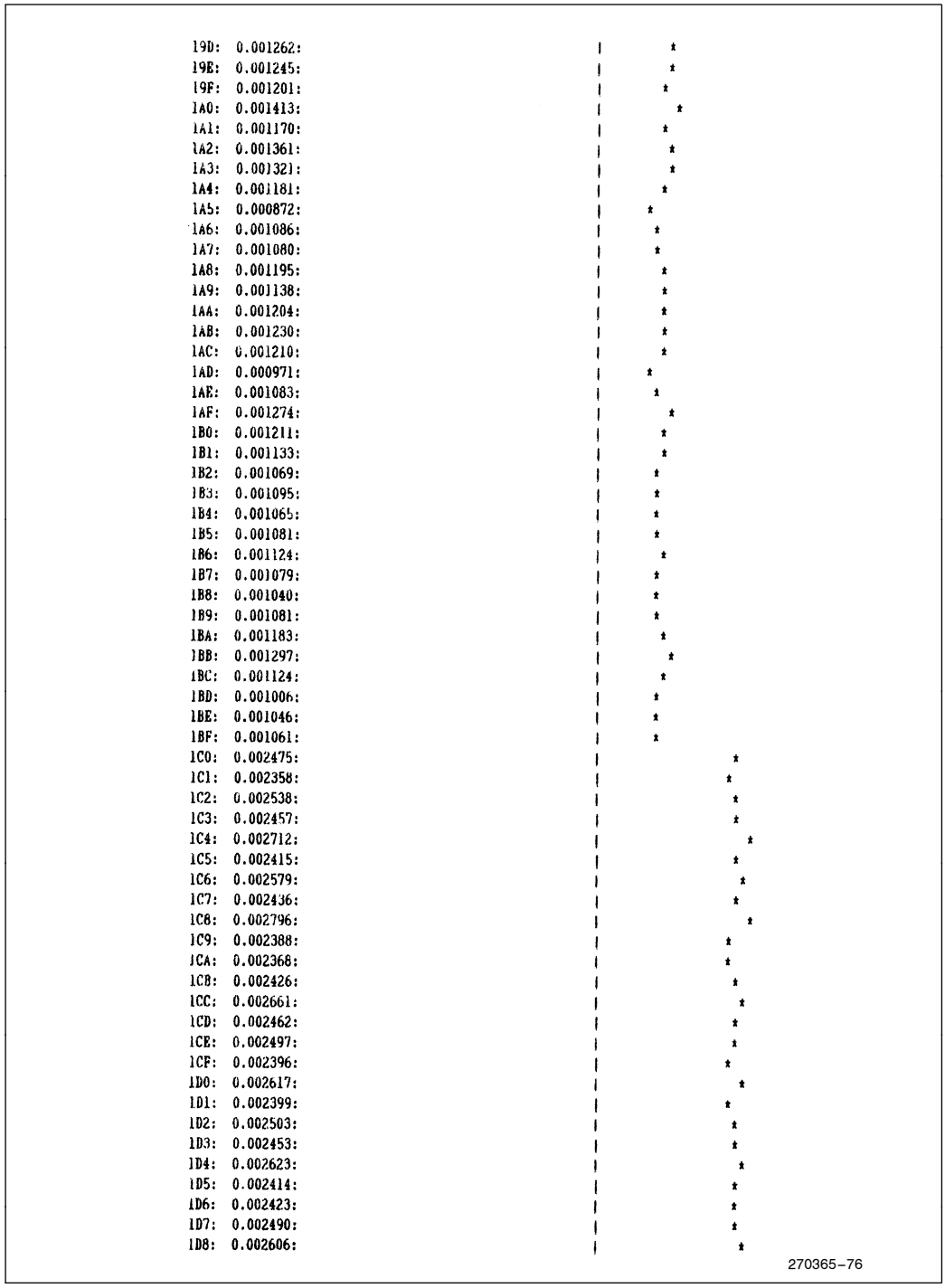
Absolute Error, SN = 4130 (Continued)



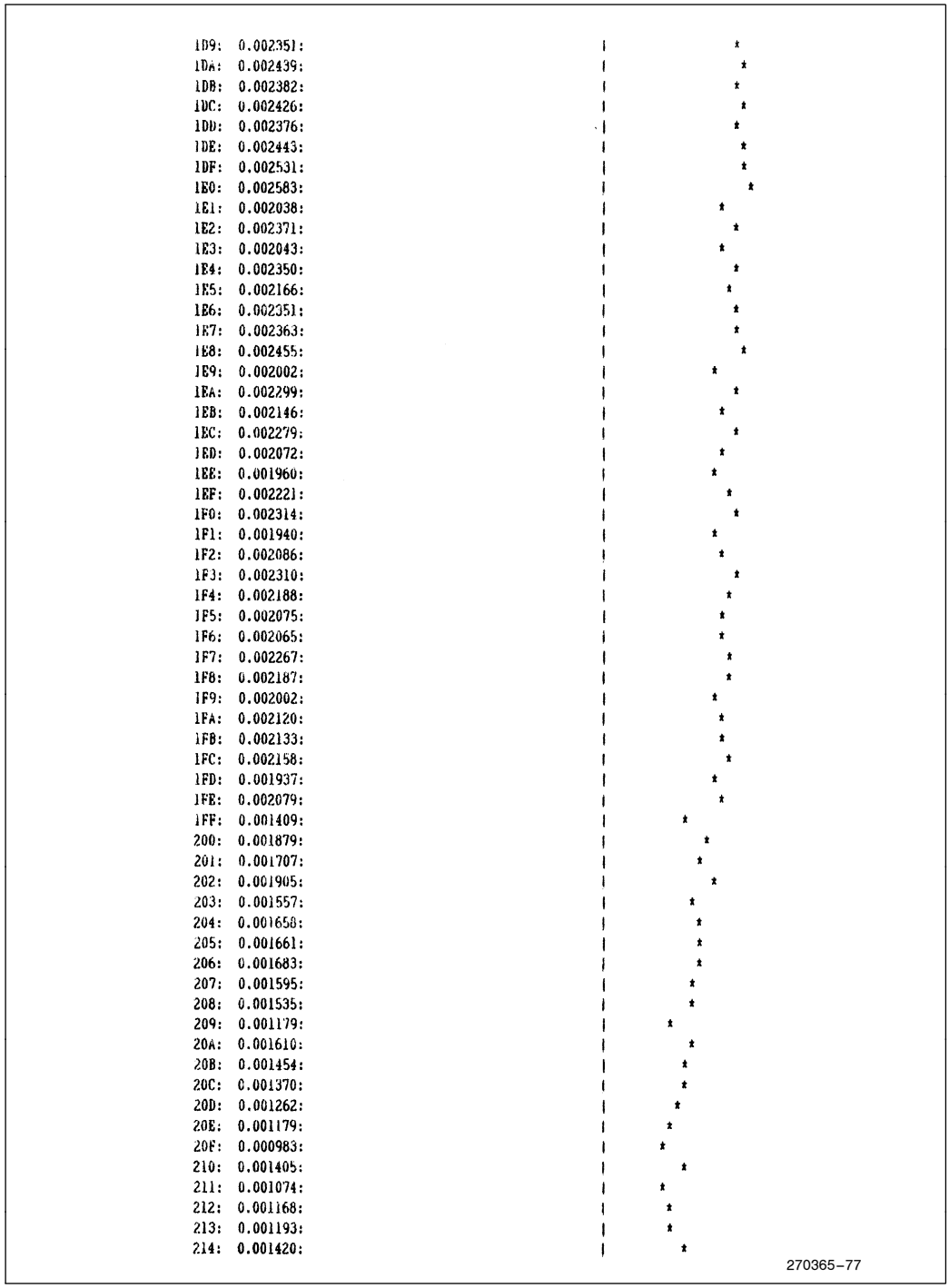
Absolute Error, SN = 4130 (Continued)



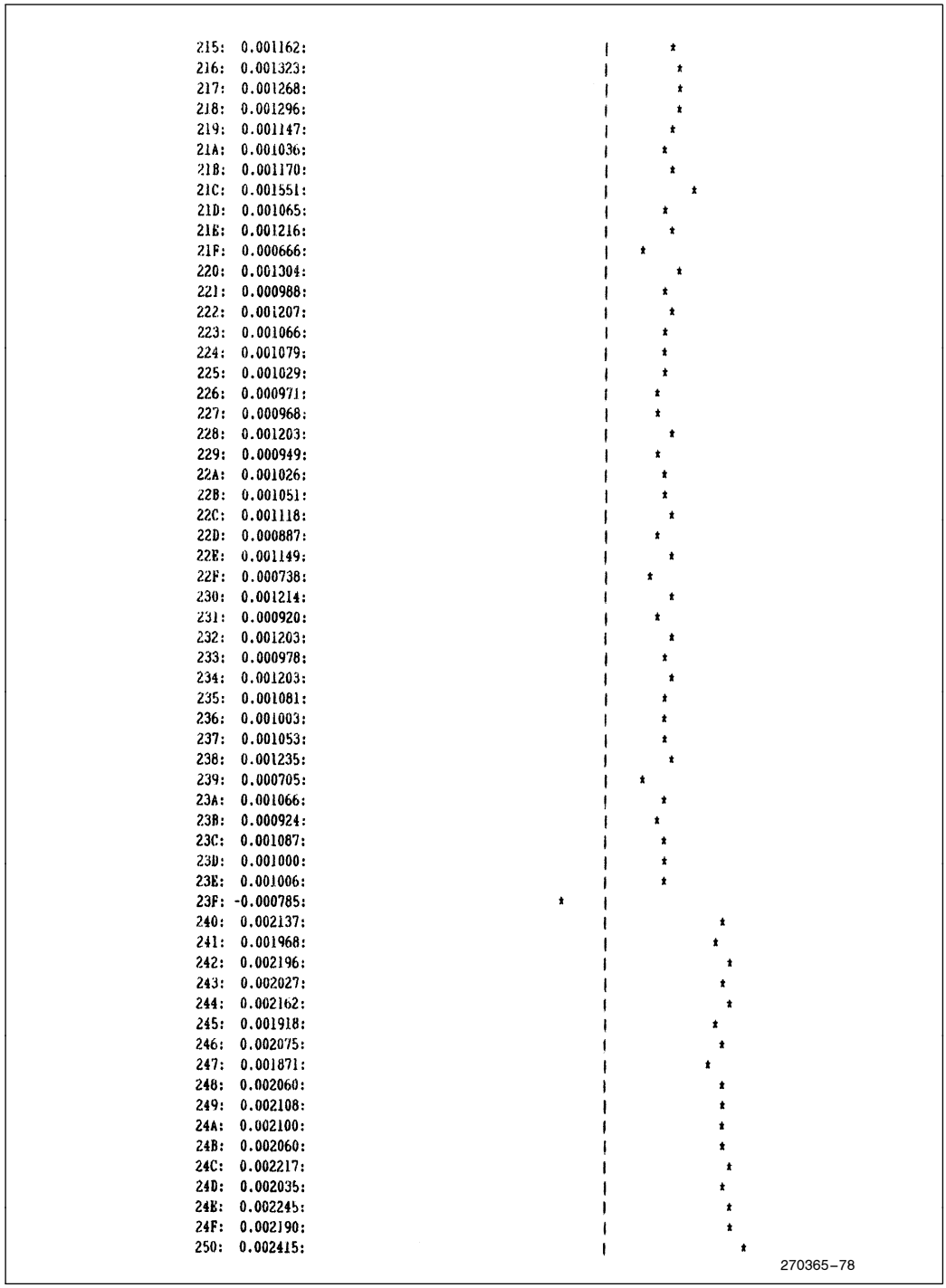
Absolute Error, SN = 4130 (Continued)



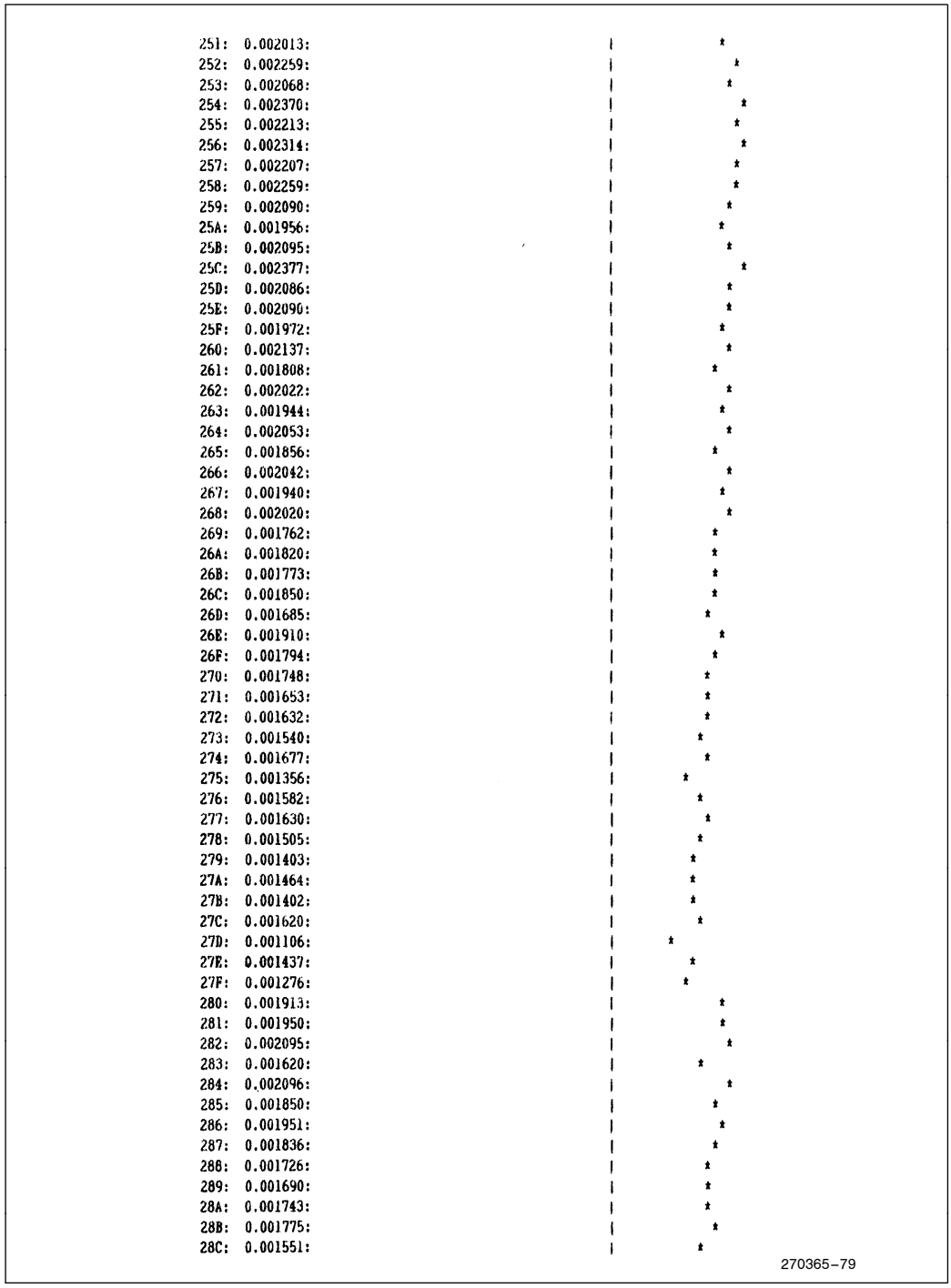
Absolute Error, SN = 4130 (Continued)



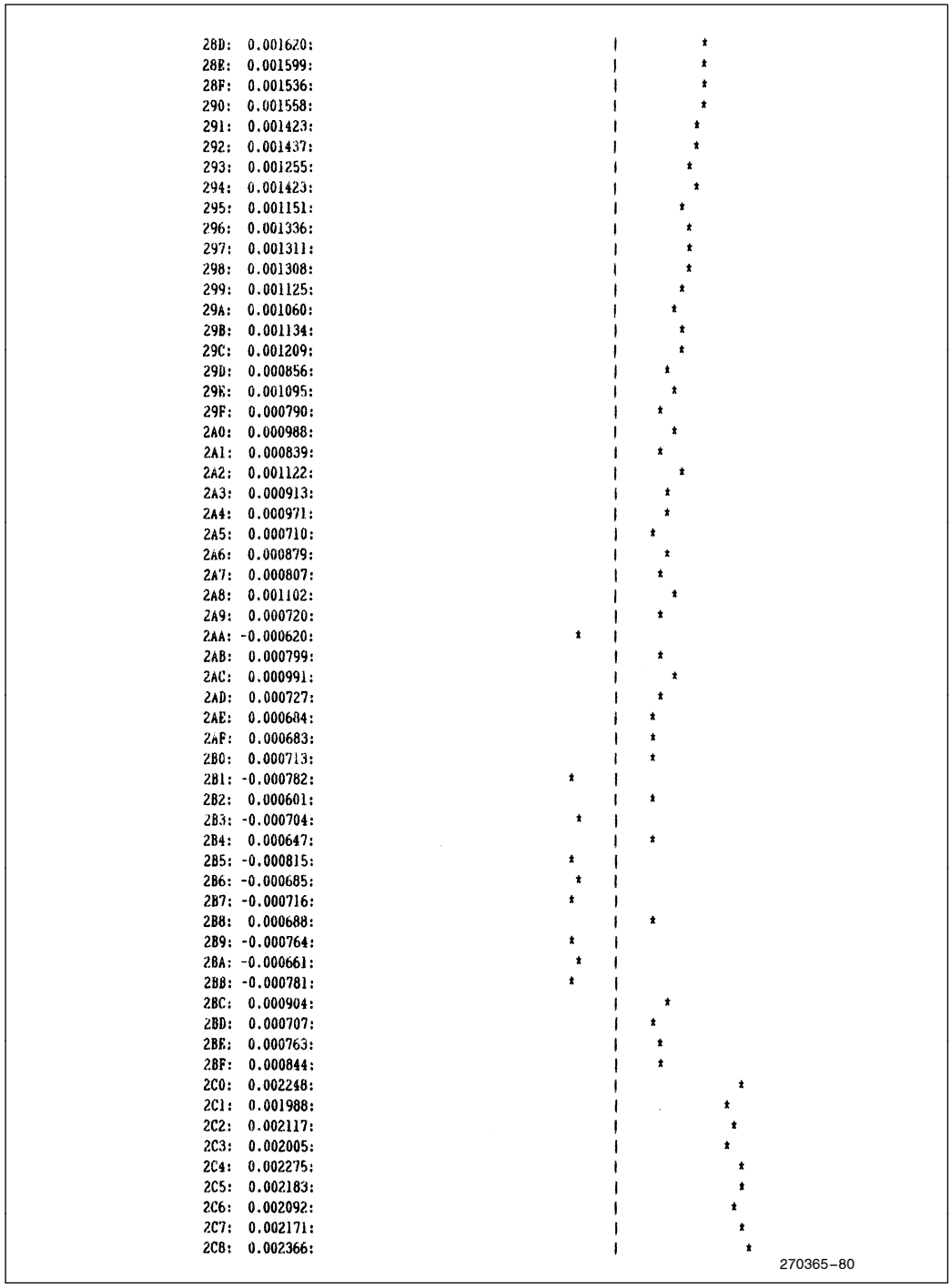
Absolute Error, SN = 4130 (Continued)



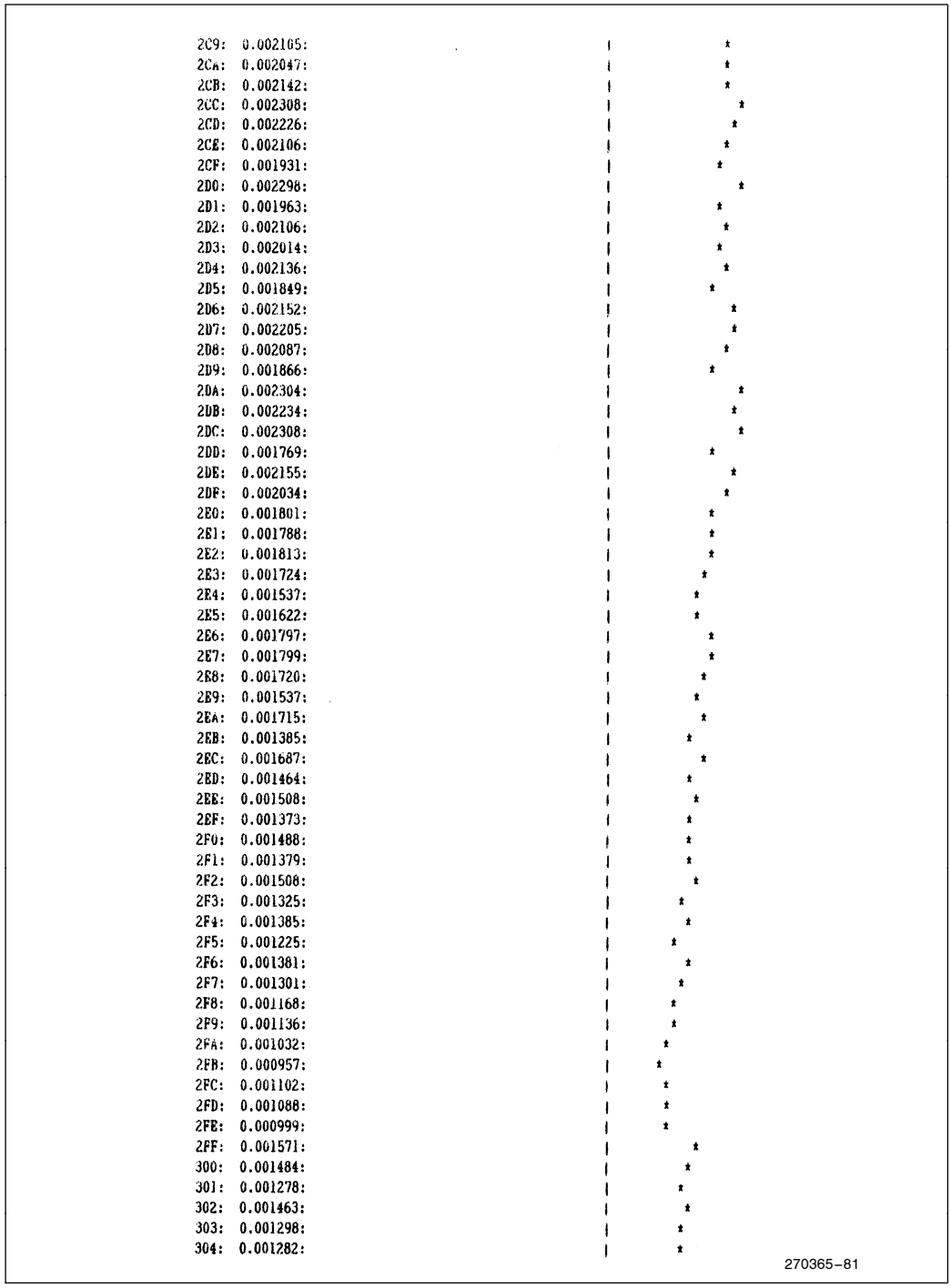
Absolute Error, SN = 4130 (Continued)



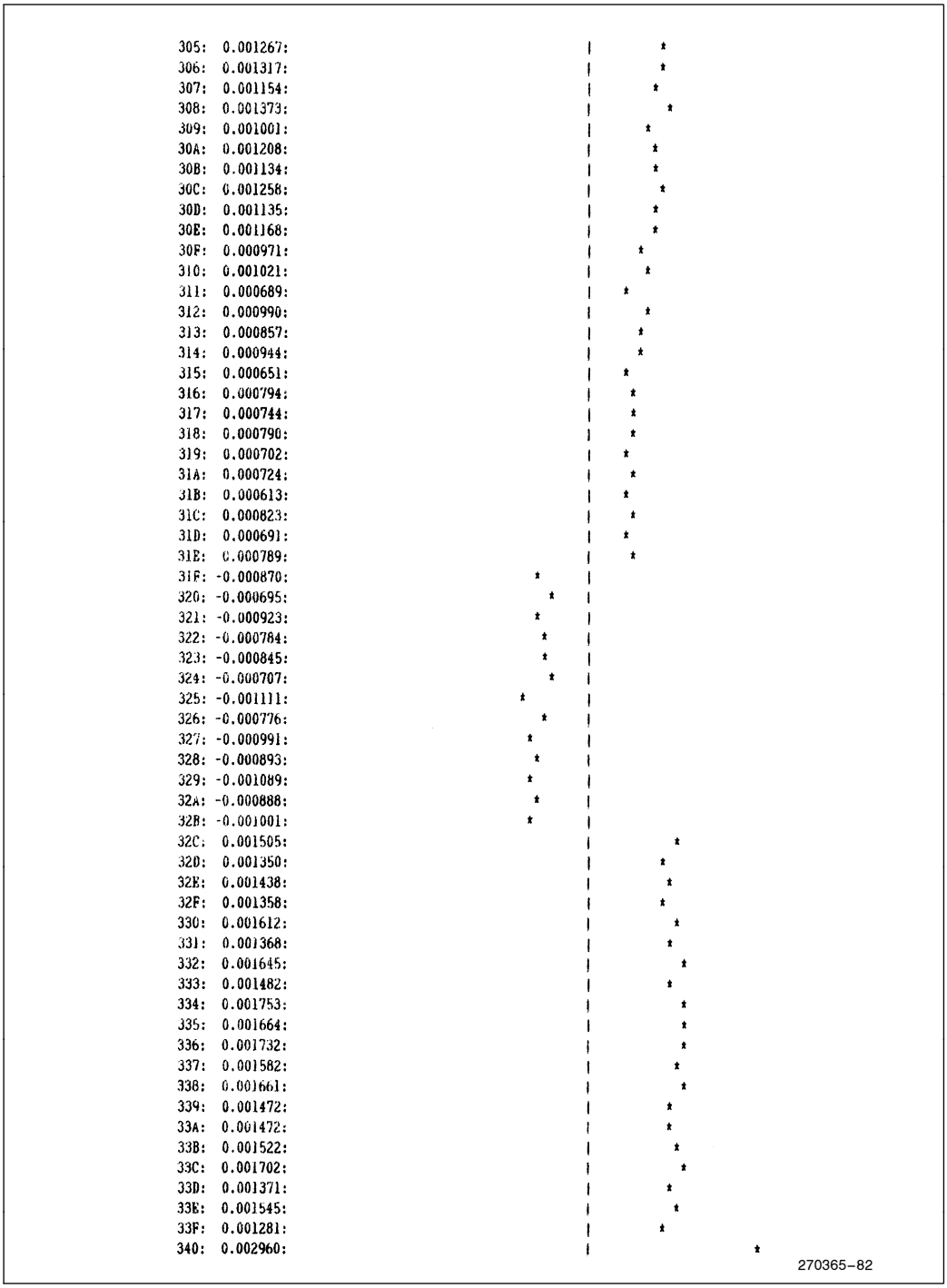
Absolute Error, SN = 4130 (Continued)



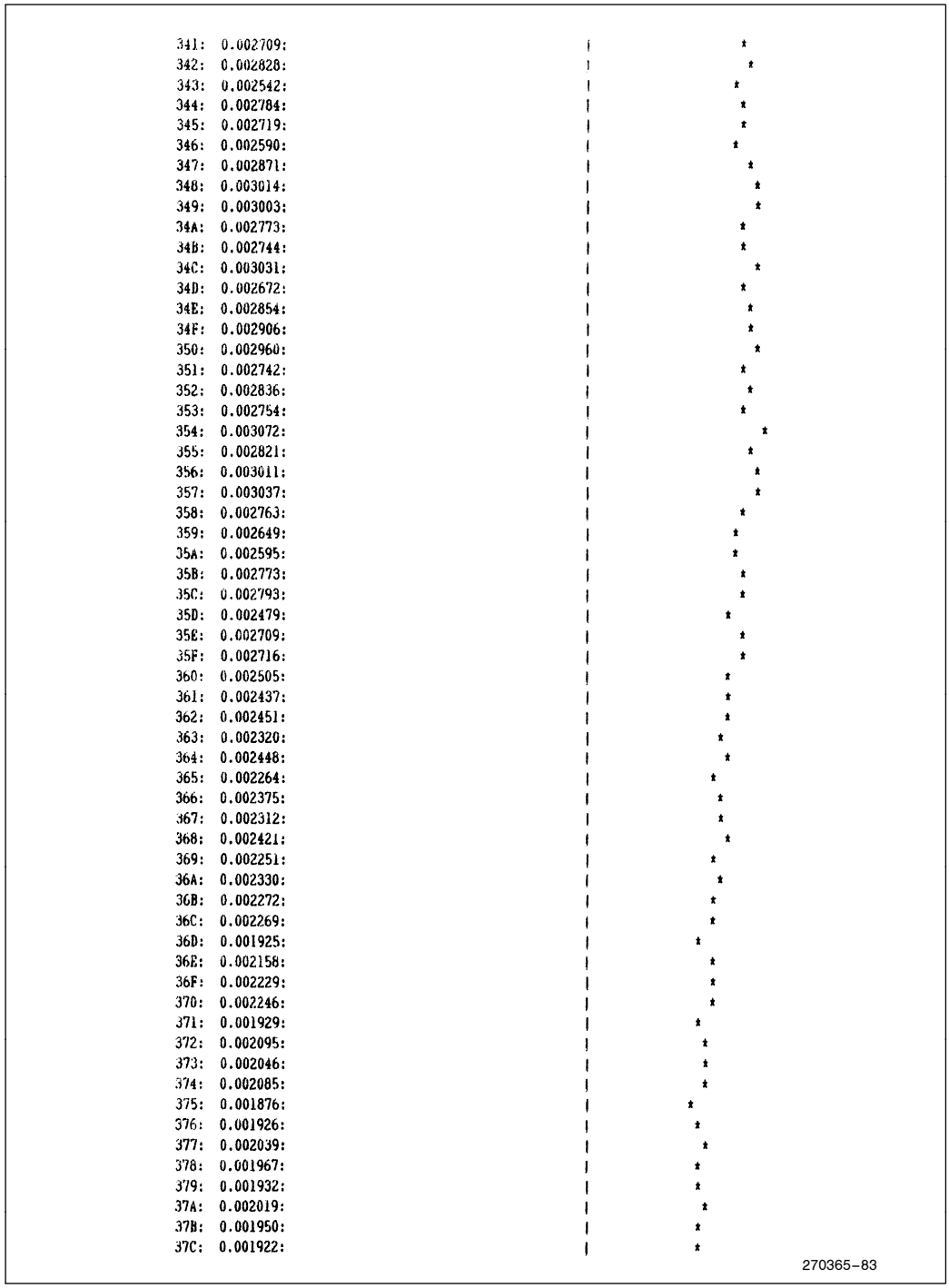
Absolute Error, SN = 4130 (Continued)



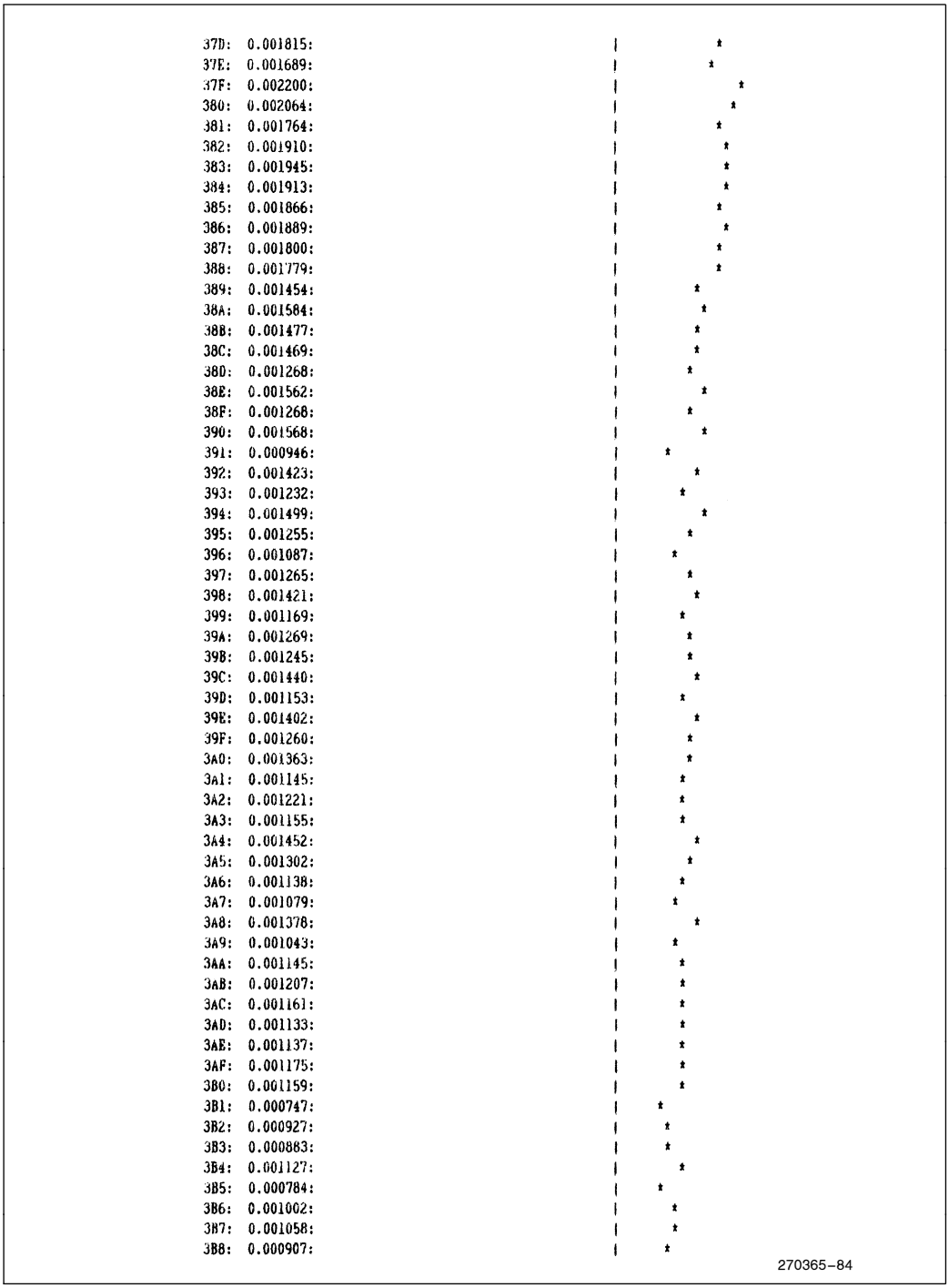
Absolute Error, SN = 4130 (Continued)



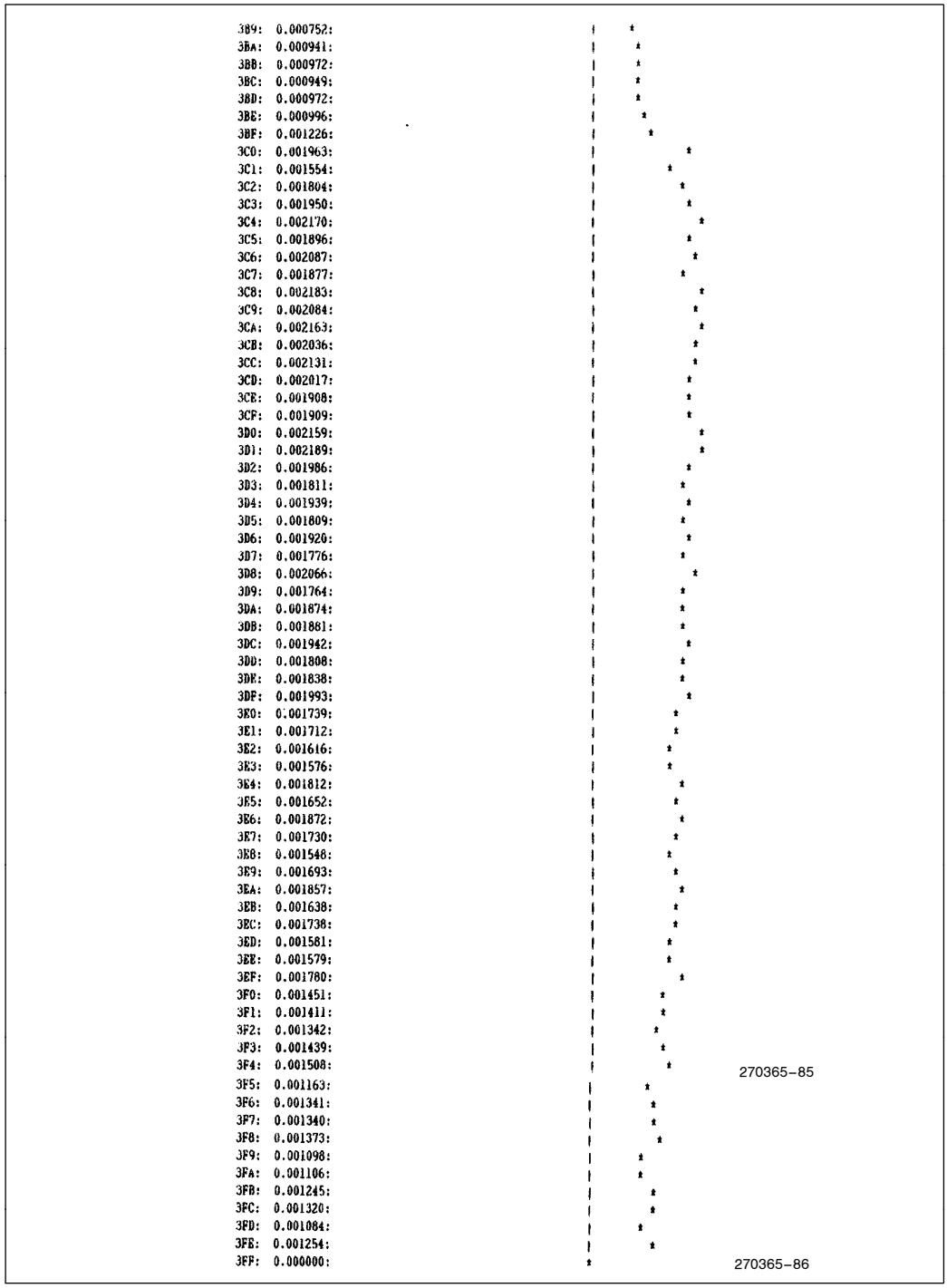
Absolute Error, SN = 4130 (Continued)



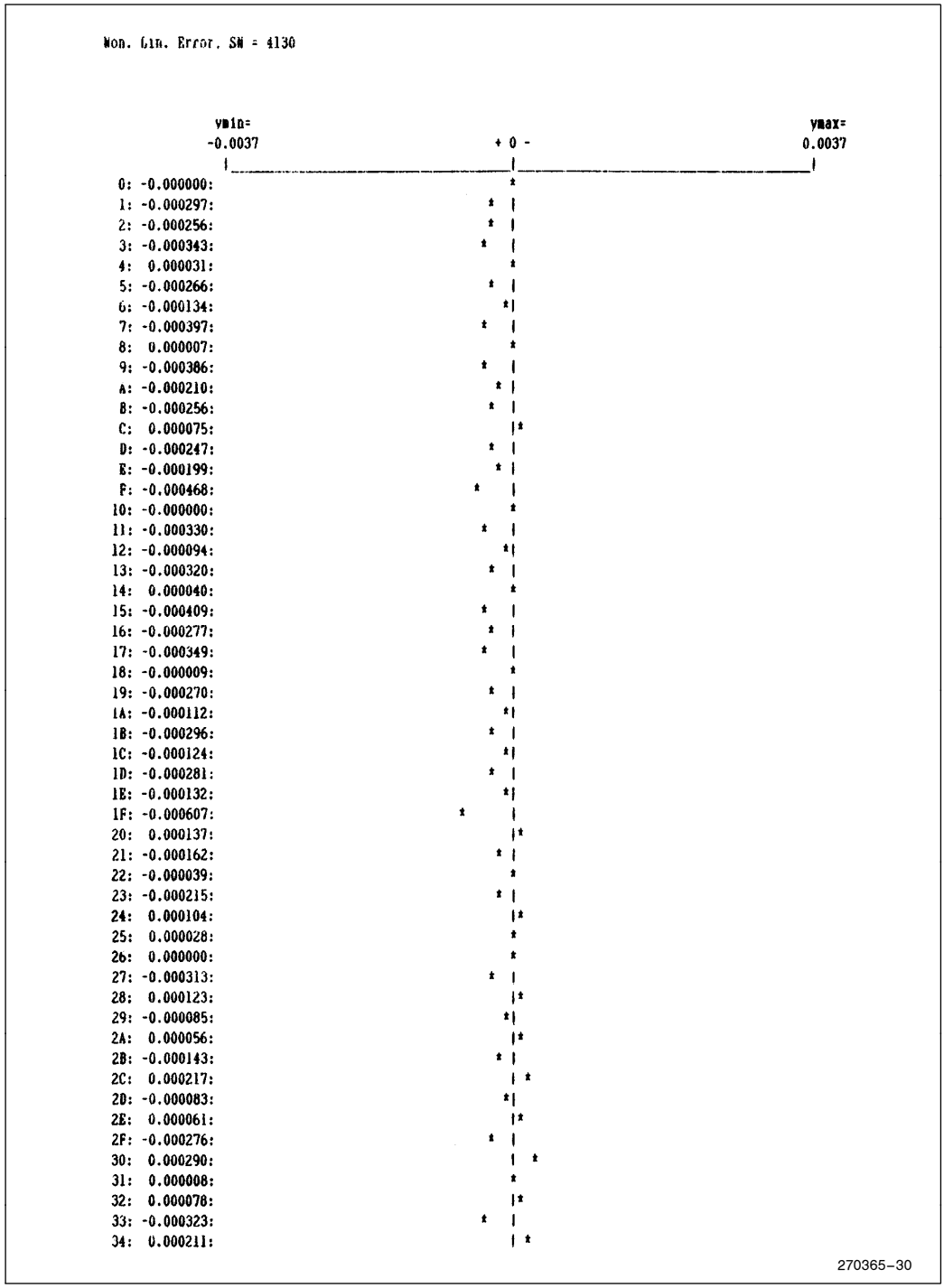
Absolute Error, SN = 4130 (Continued)



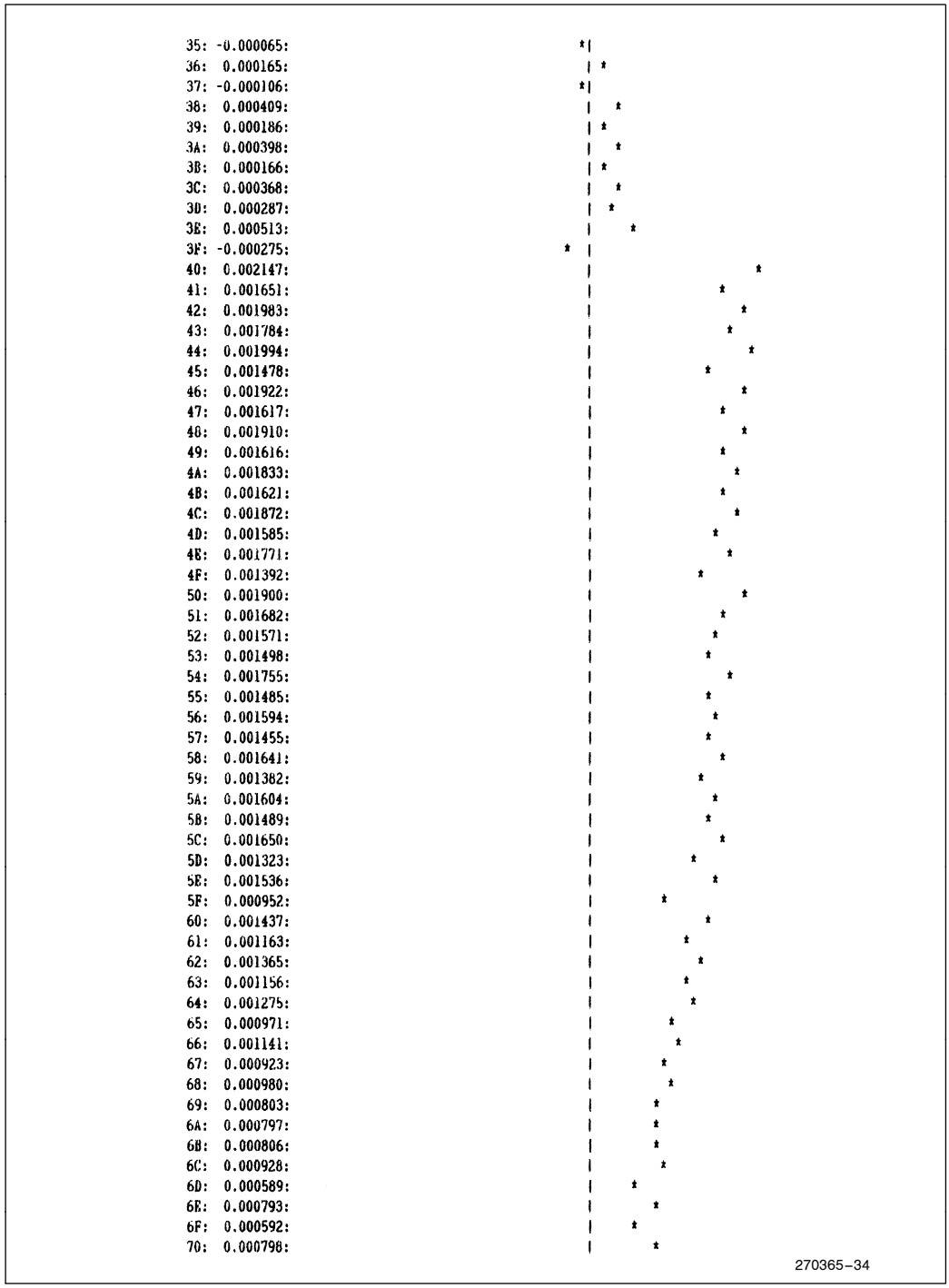
Absolute Error, SN = 4130 (Continued)



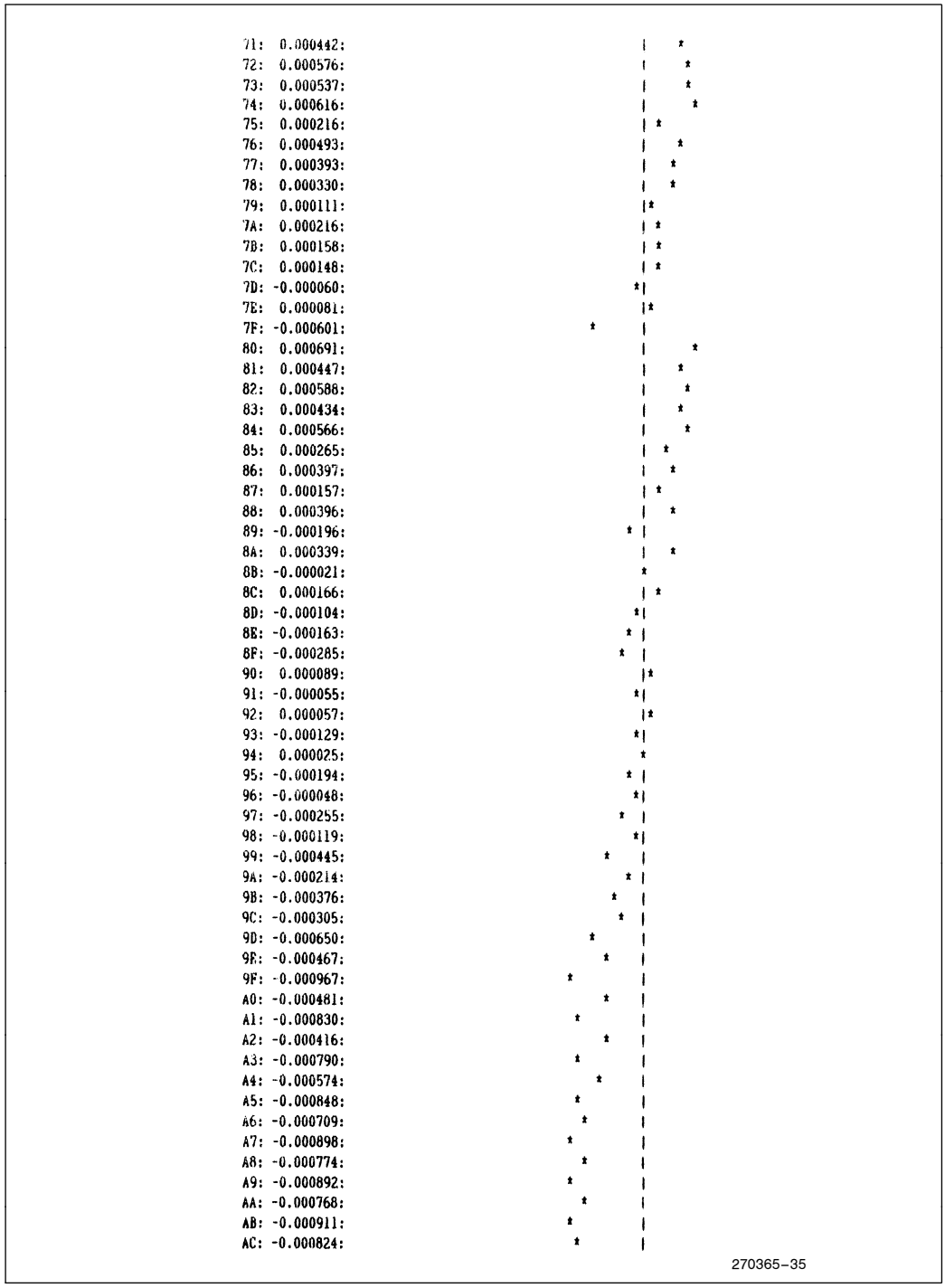
Absolute Error, SN = 4130 (Continued)



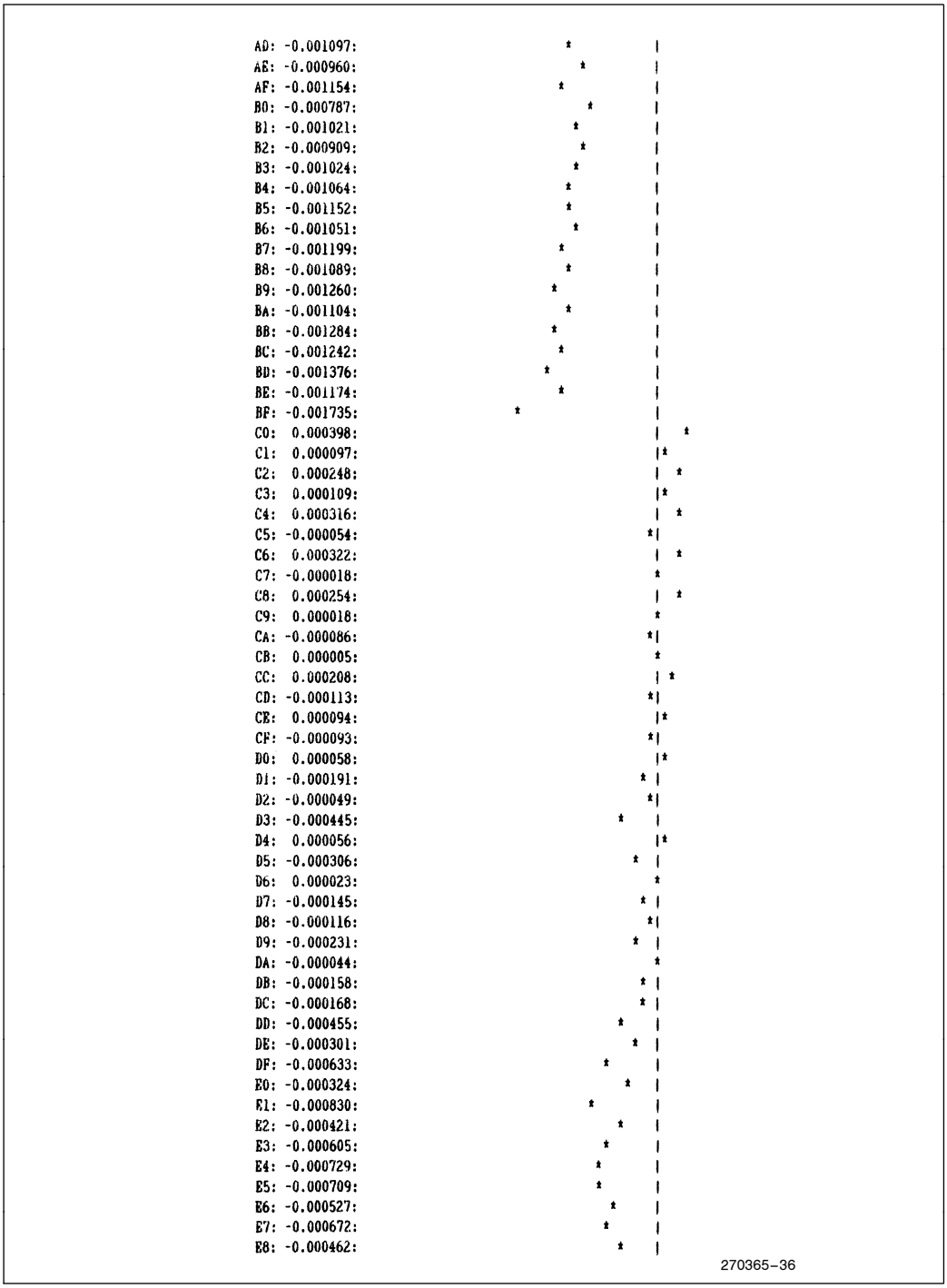
Non. Lin. Error, SN = 4130



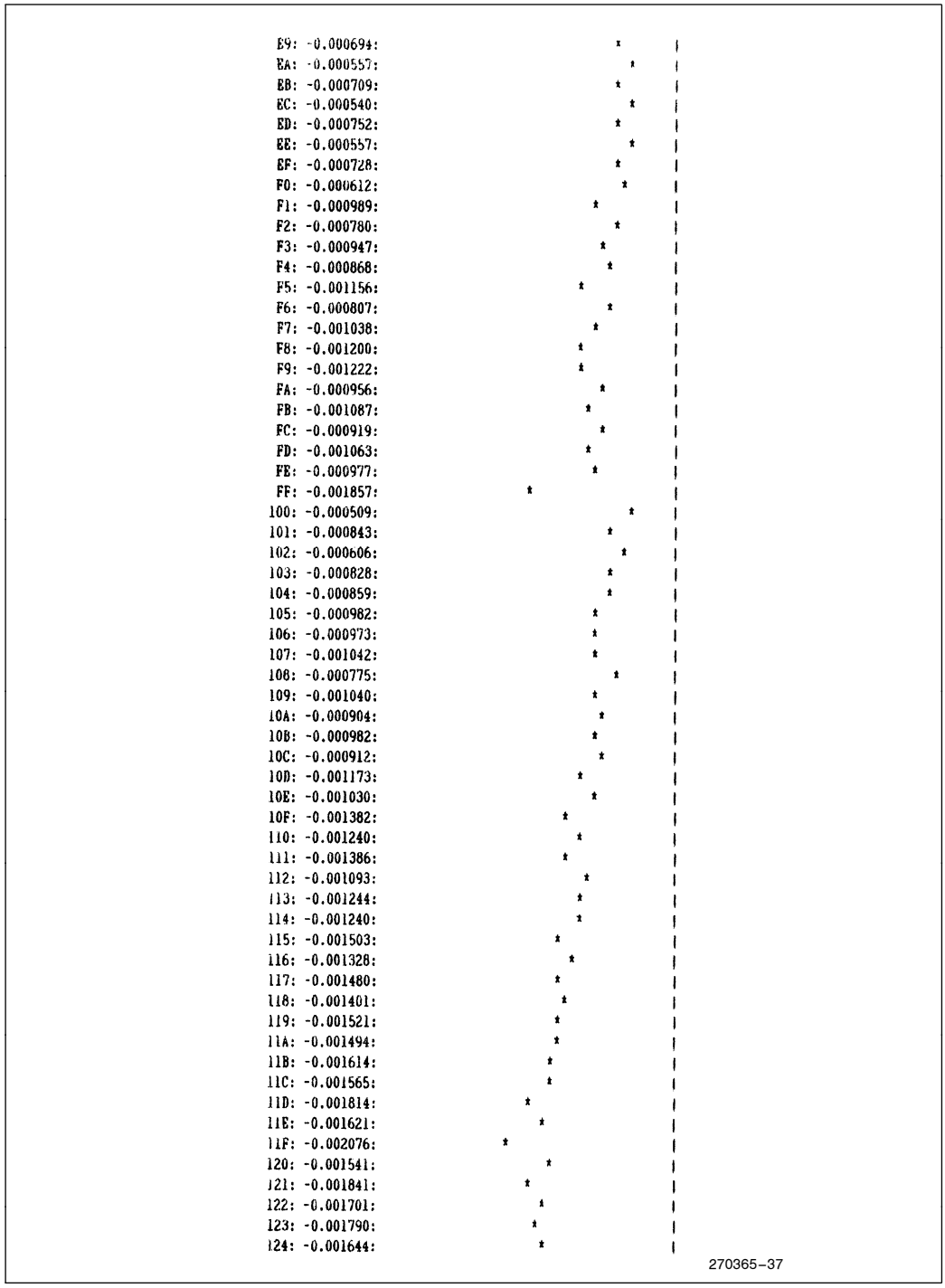
Non. Lin. Error, SN = 4130 (Continued)



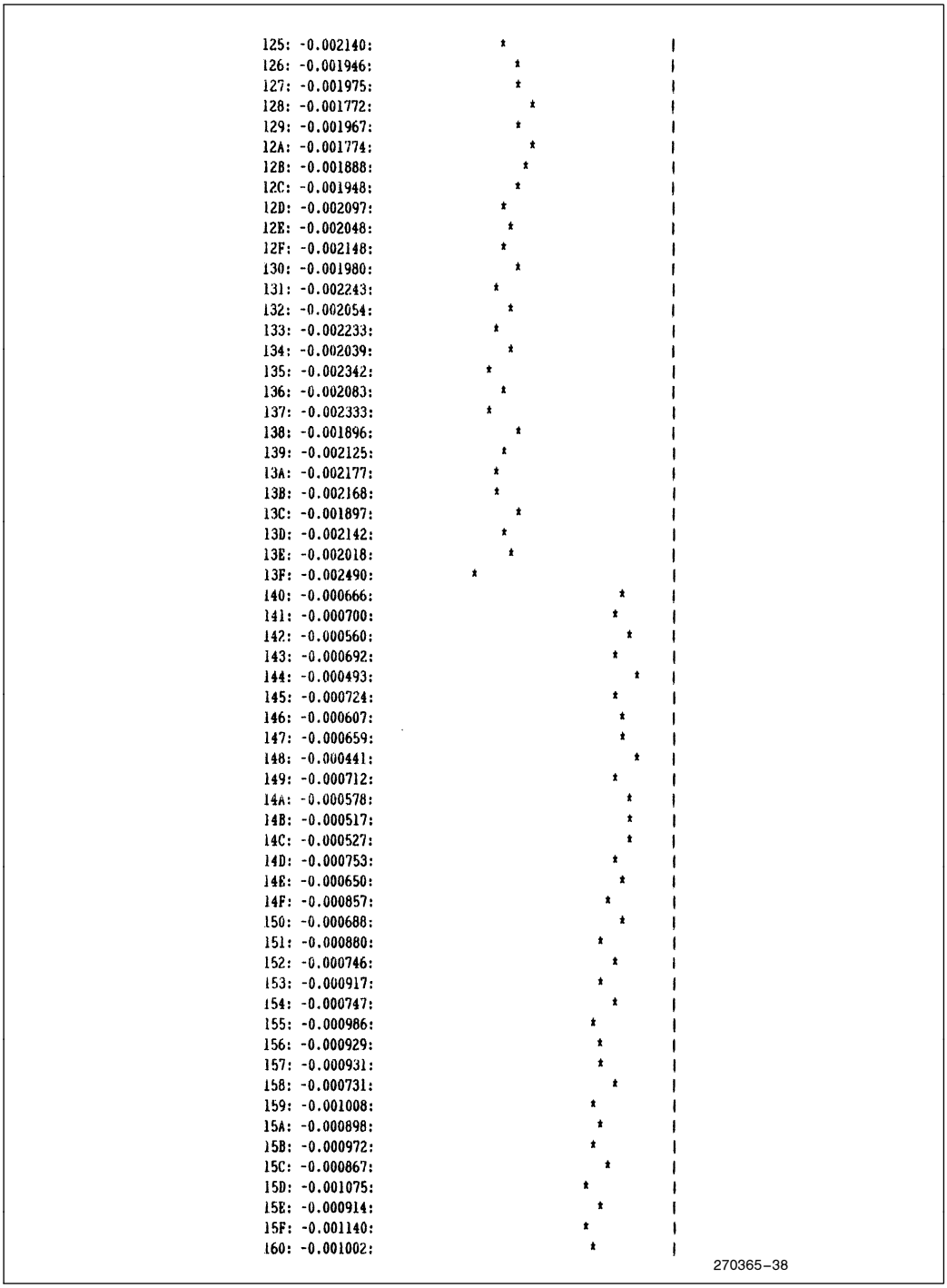
Non. Lin. Error, SN = 4130 (Continued)



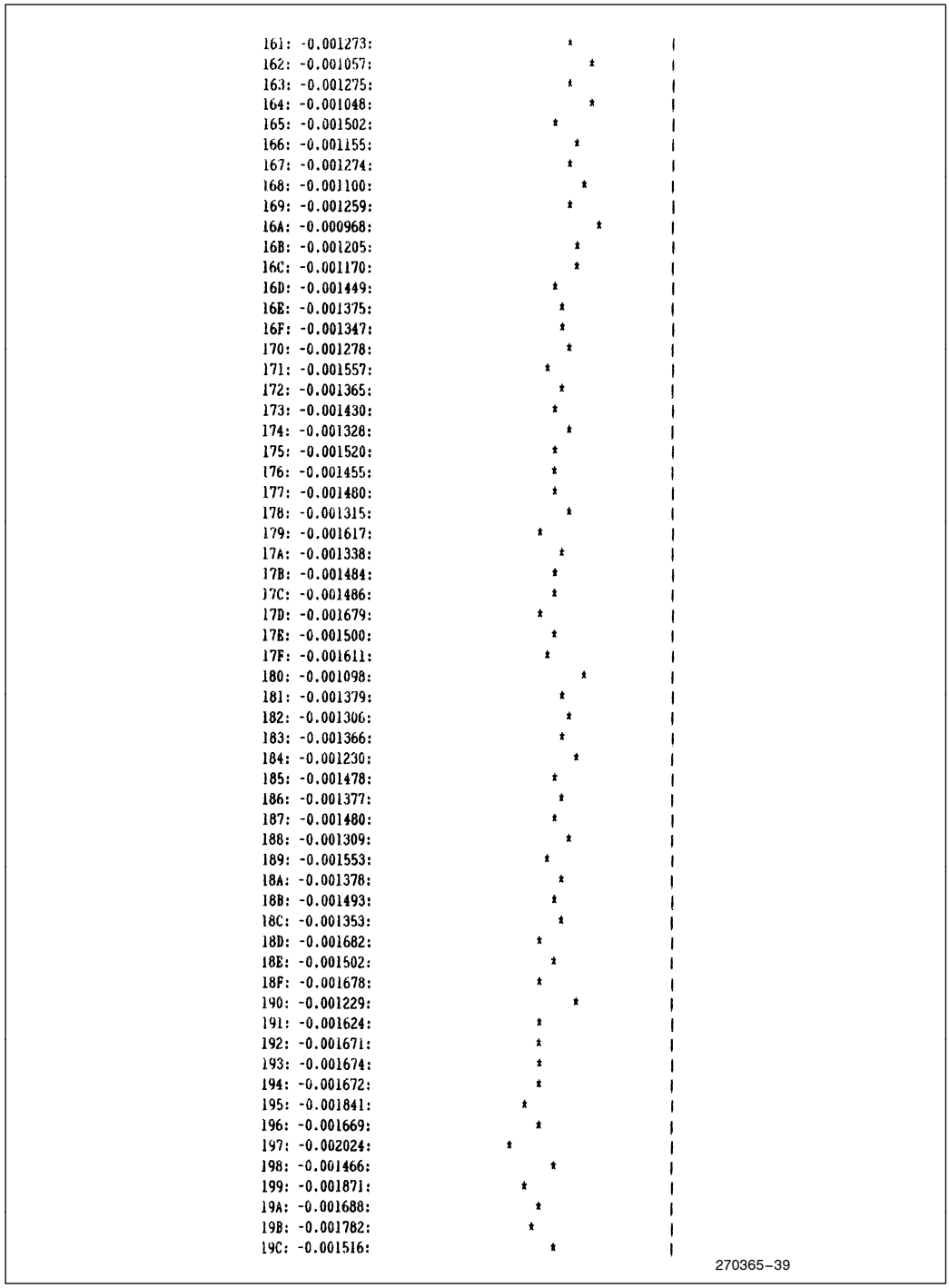
Non. Lin. Error, SN = 4130 (Continued)



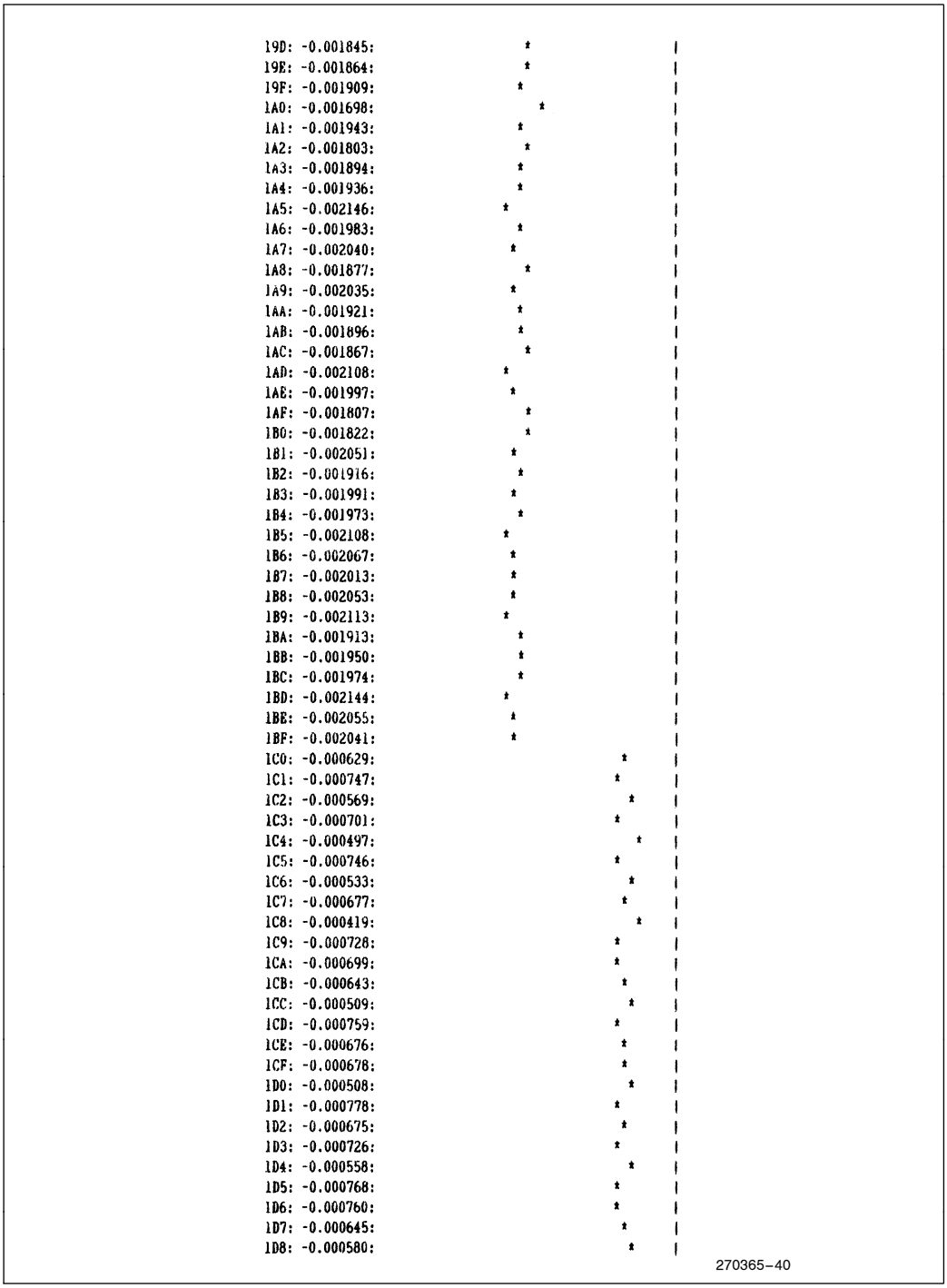
Non. Lin. Error, SN = 4130 (Continued)



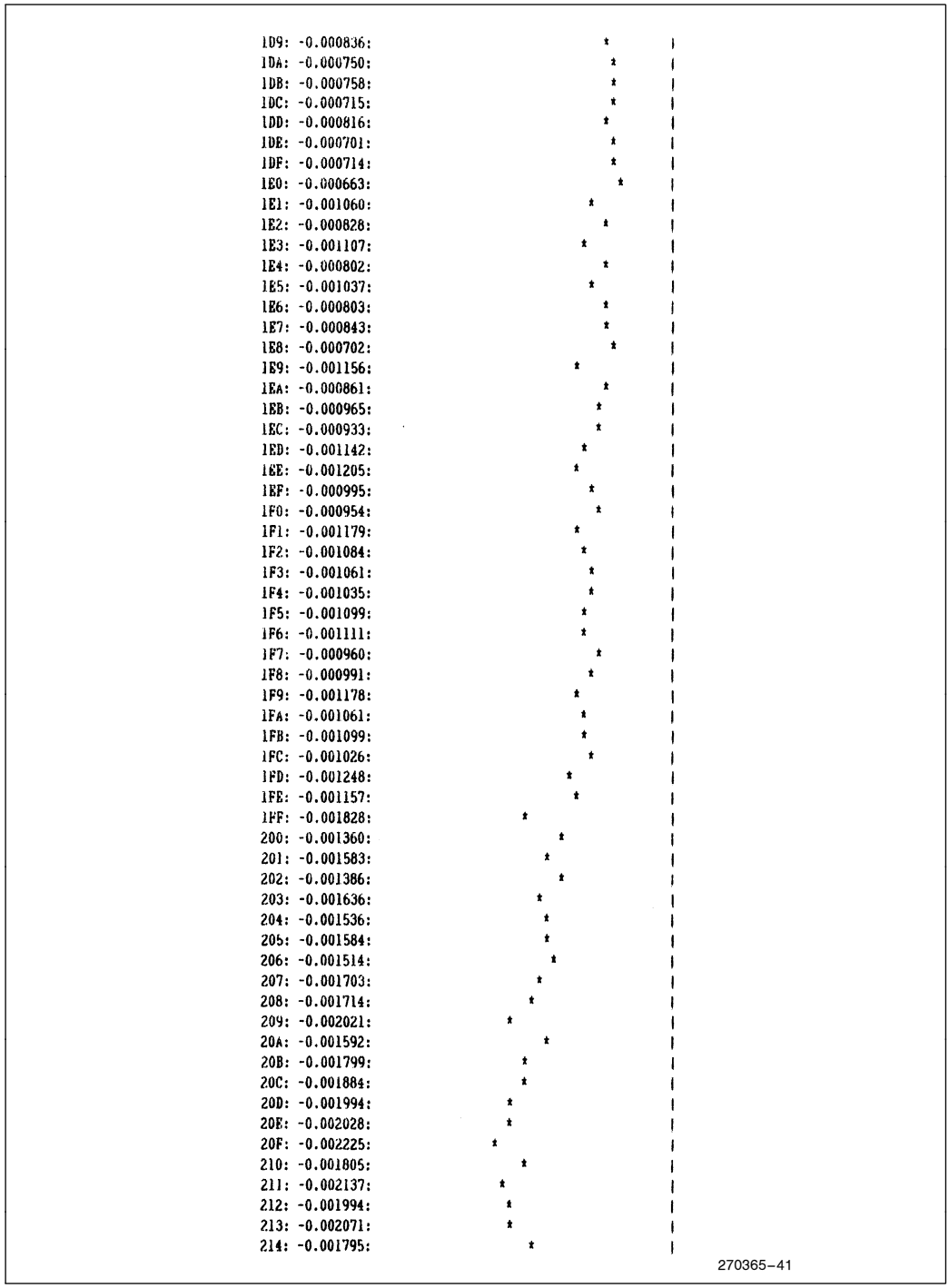
Non. Lin. Error, SN = 4130 (Continued)



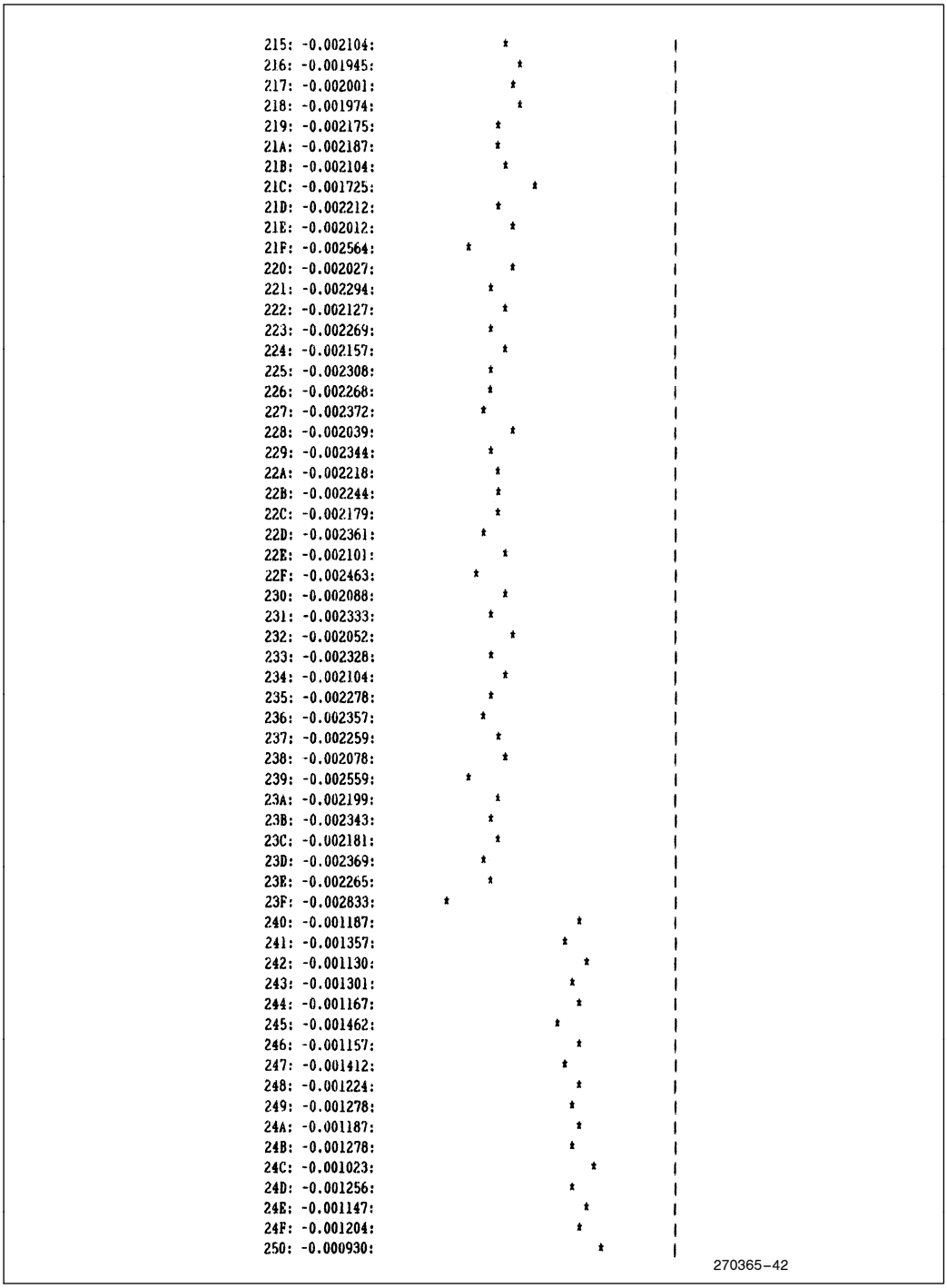
Non. Lin. Error, SN = 4130 (Continued)



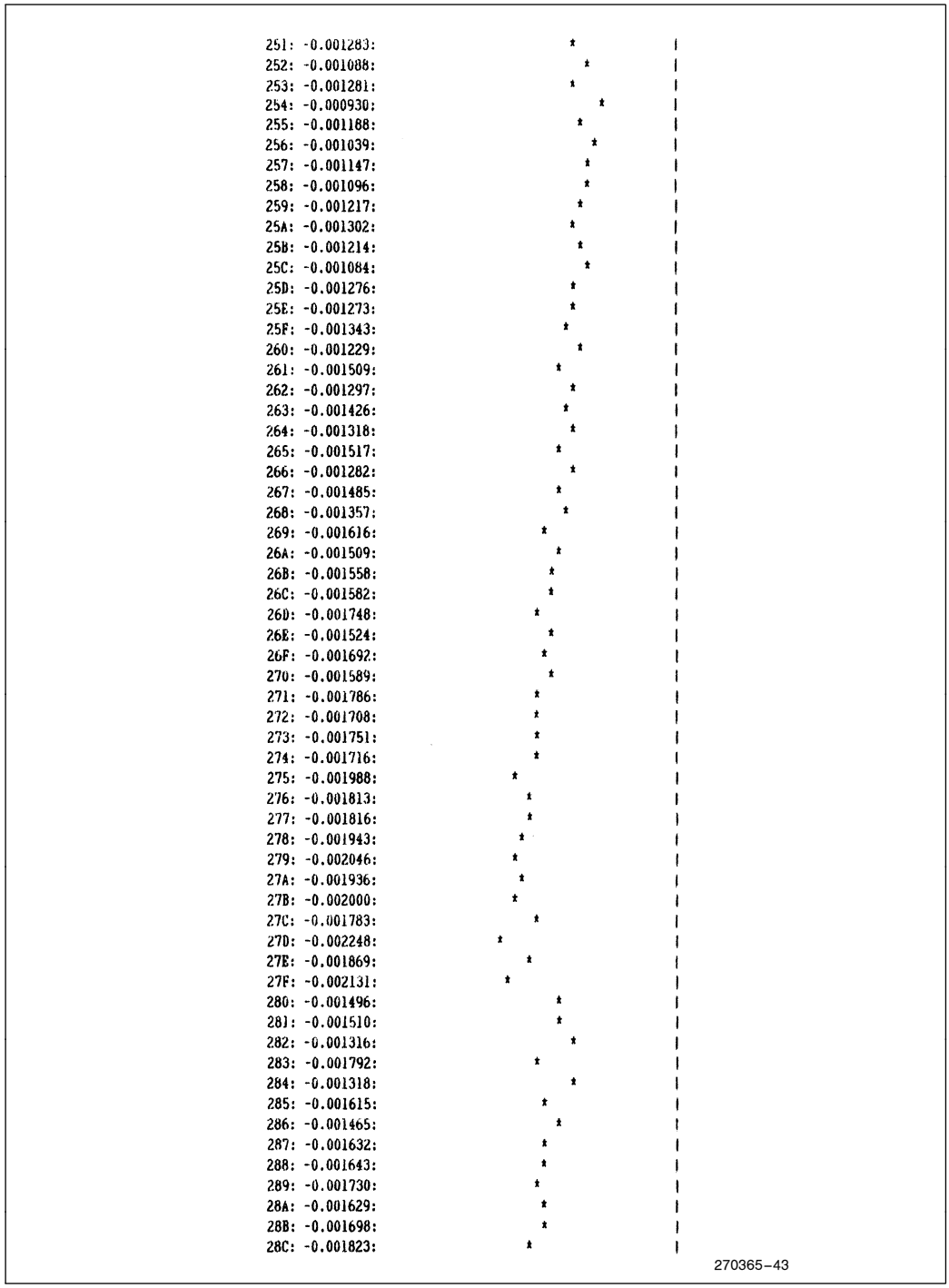
Non. Lin. Error, SN = 4130 (Continued)



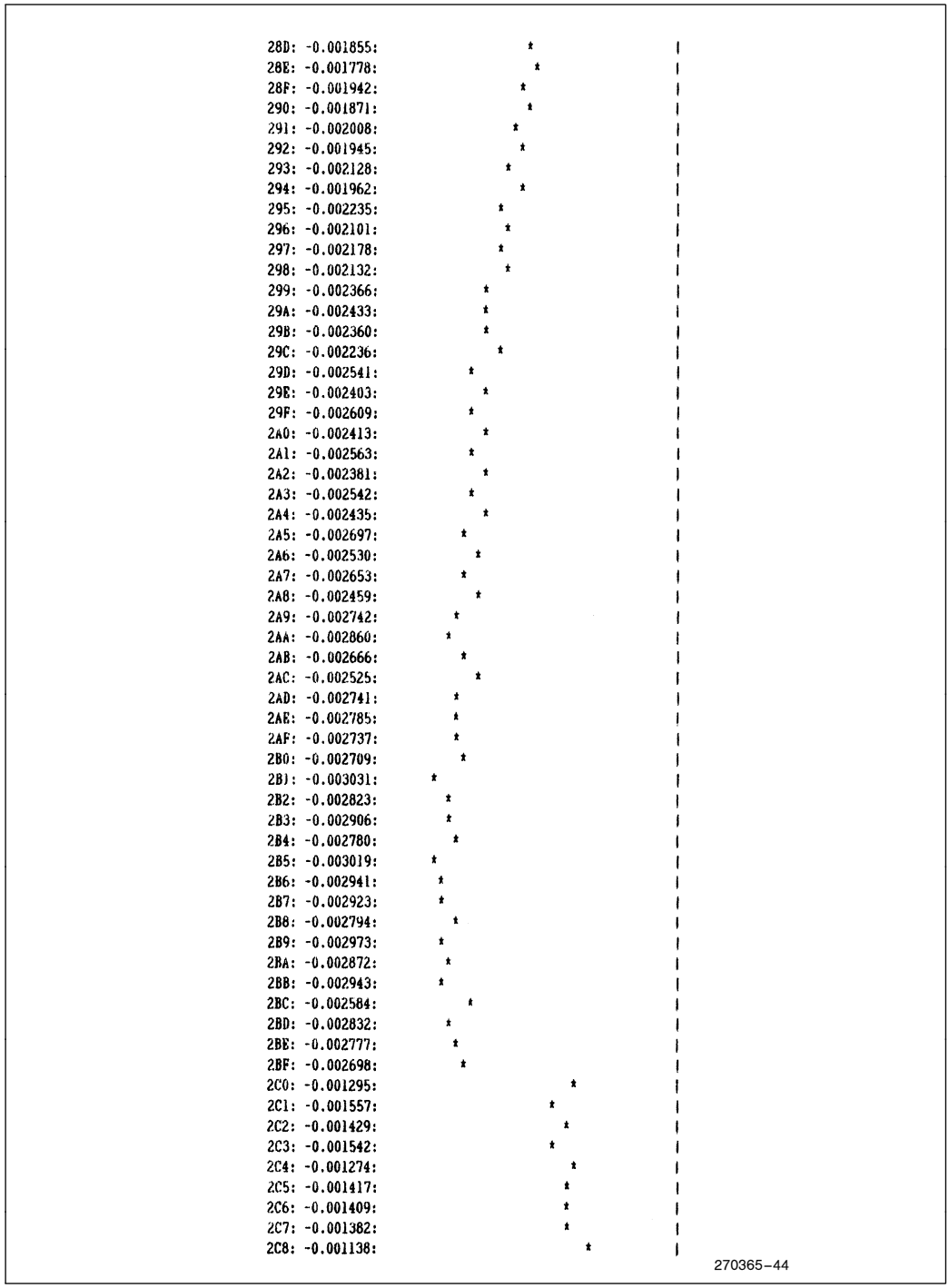
Non. Lin. Error, SN = 4130 (Continued)



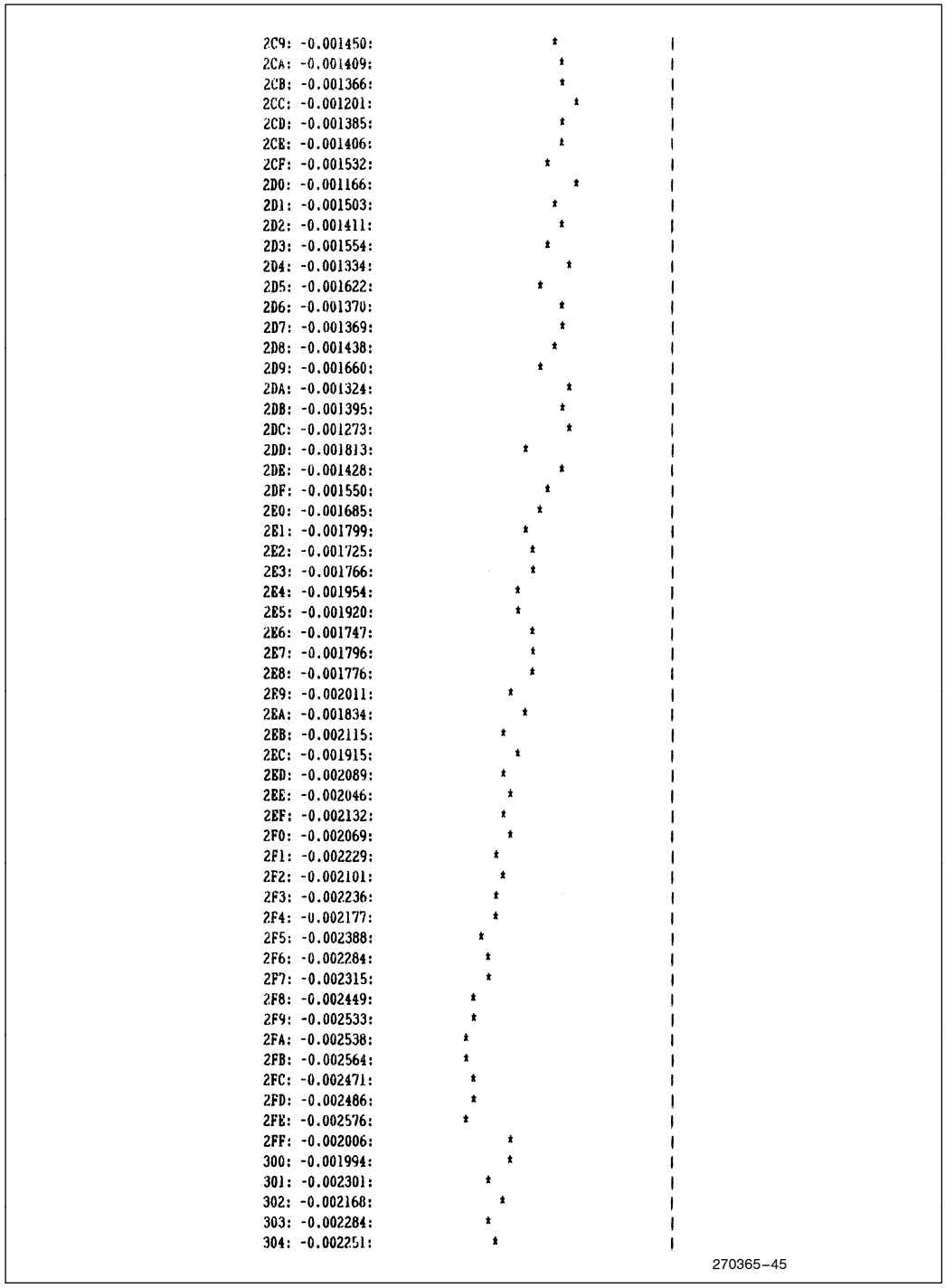
Non. Lin. Error, SN = 4130 (Continued)



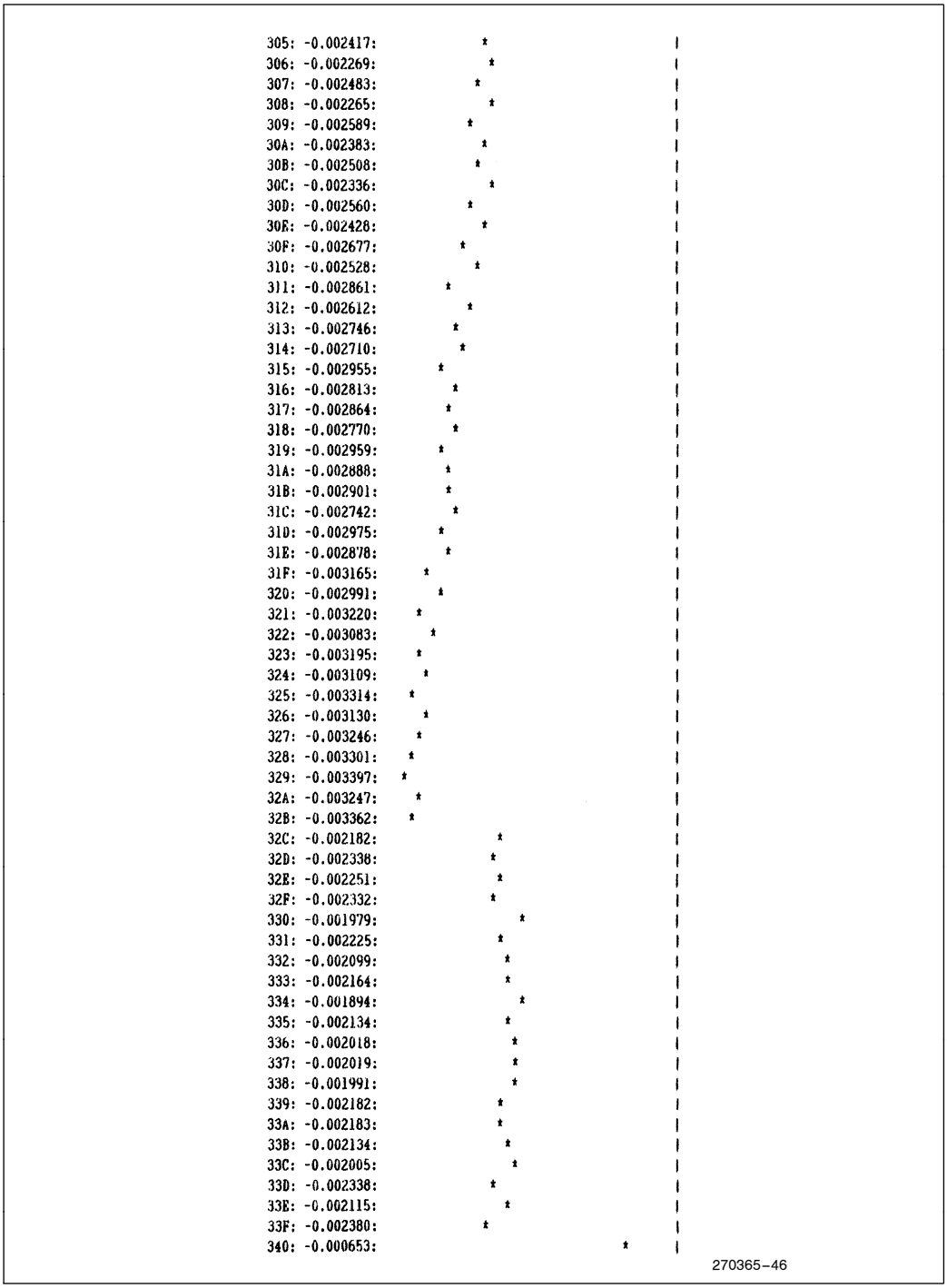
Non. Lin. Error, SN = 4130 (Continued)



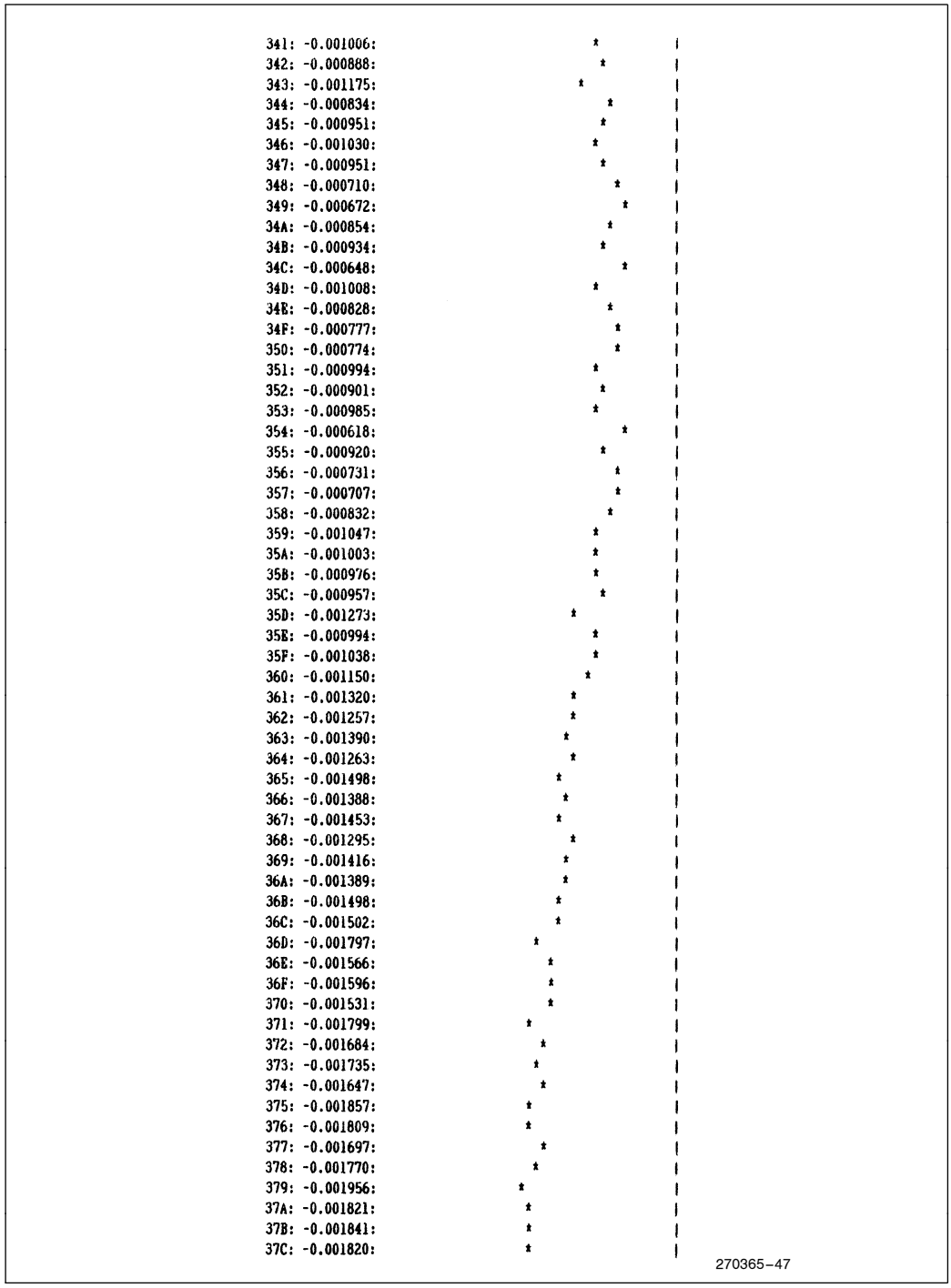
Non. Lin. Error, SN = 4130 (Continued)



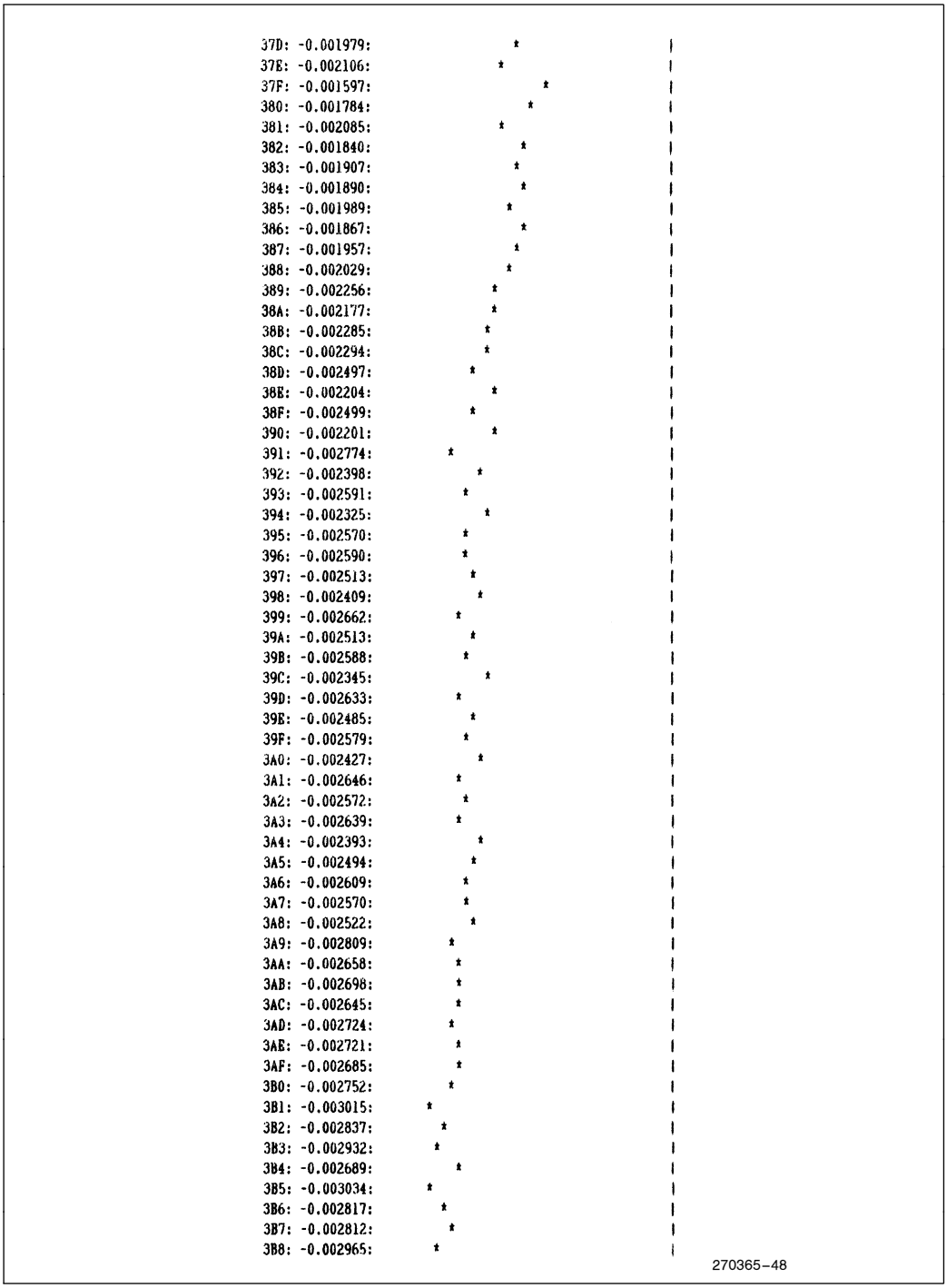
Non. Lin. Error, SN = 4130 (Continued)



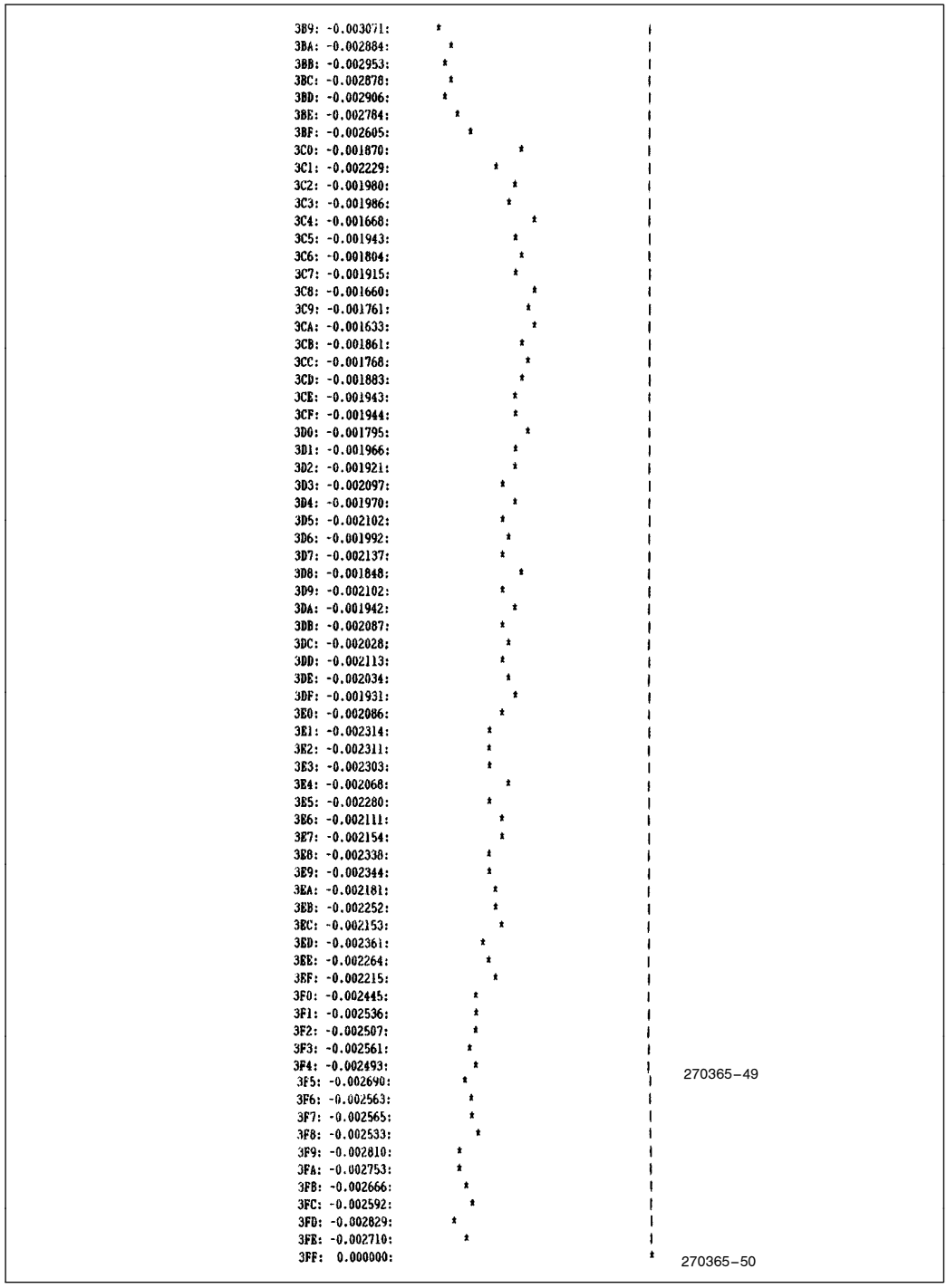
Non. Lin. Error, SN = 4130 (Continued)



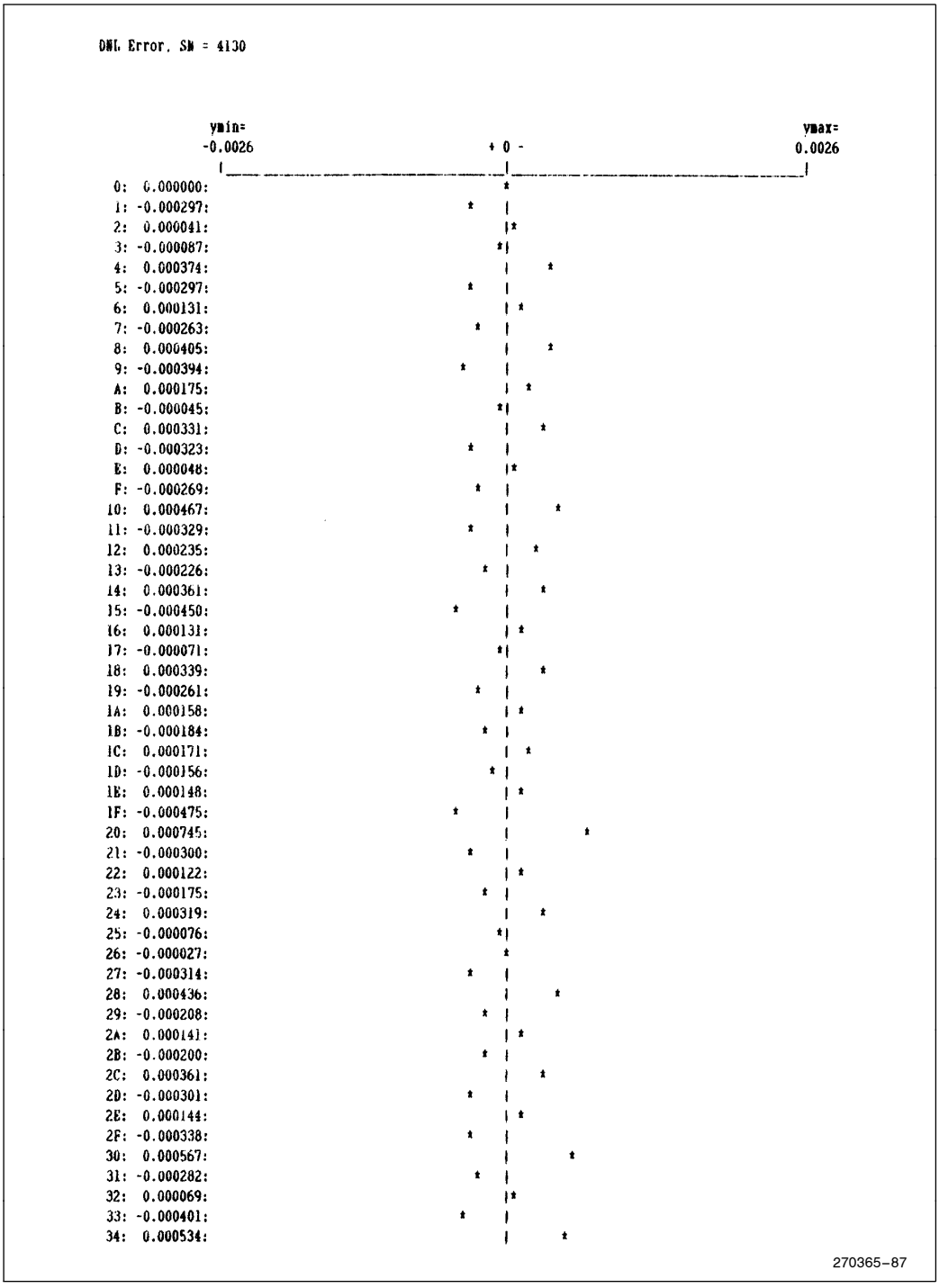
Non. Lin. Error, SN = 4130 (Continued)



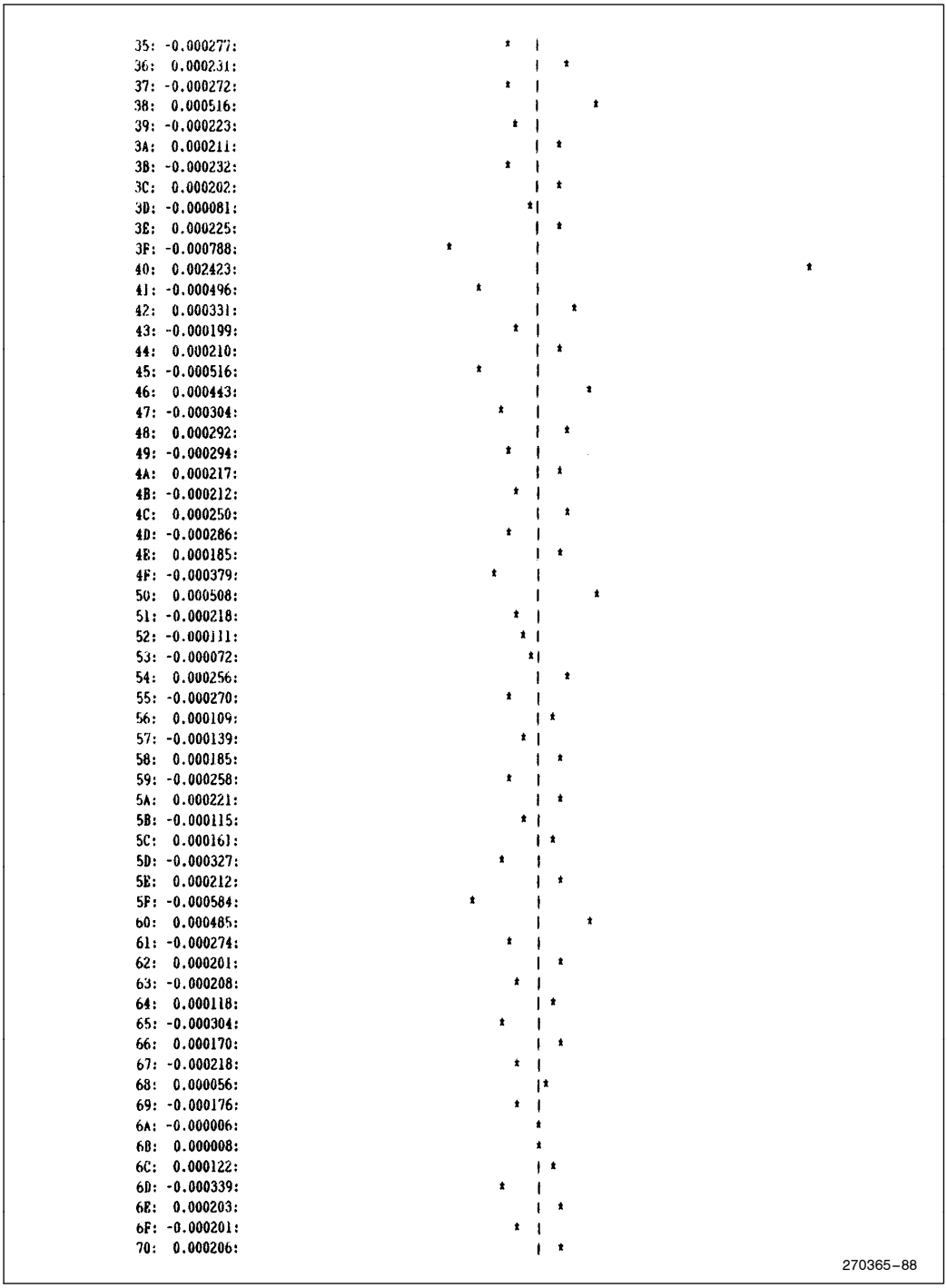
Non. Lin. Error, SN = 4130 (Continued)



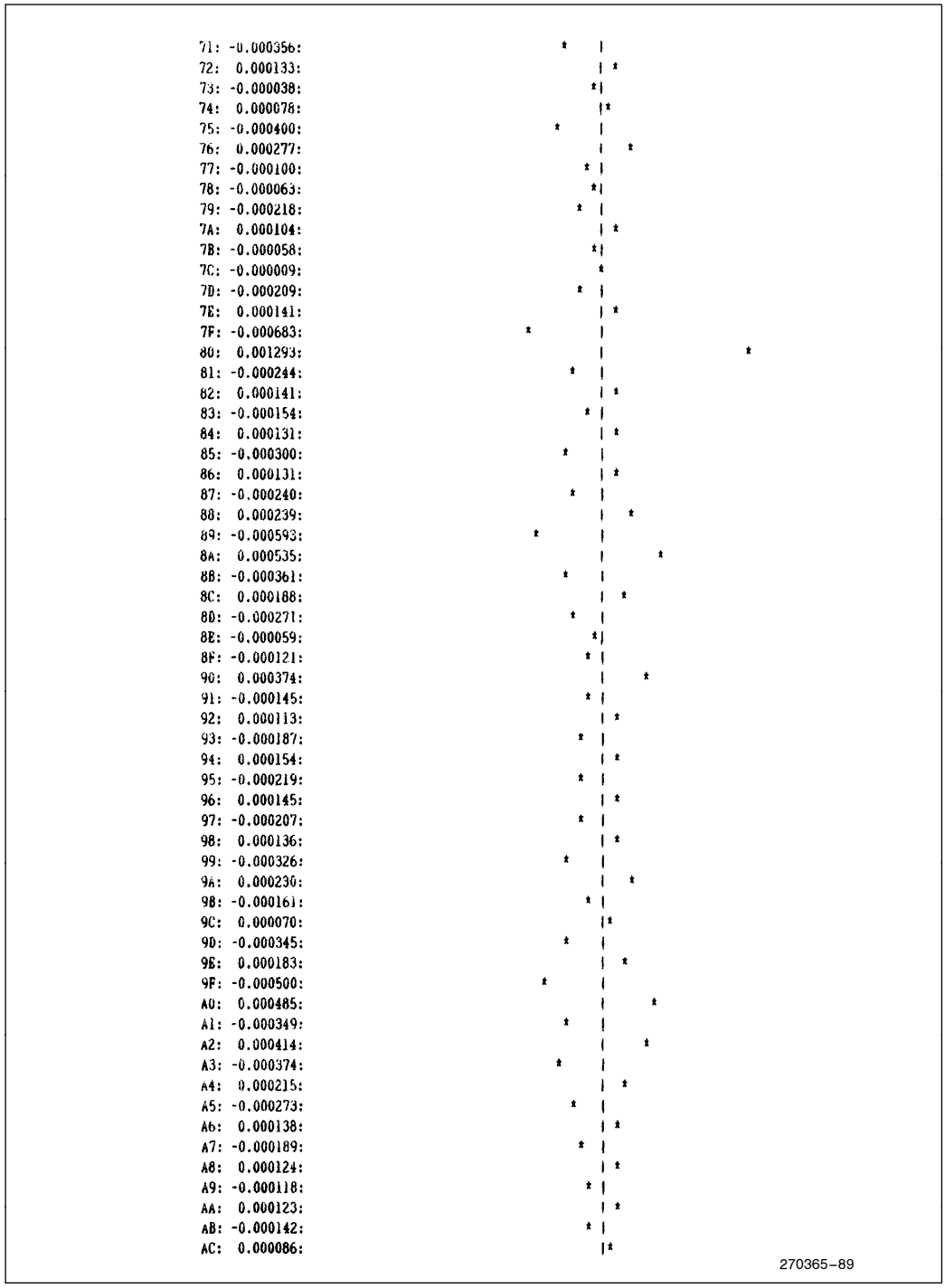
Non. Lin. Error, SN = 4130 (Continued)



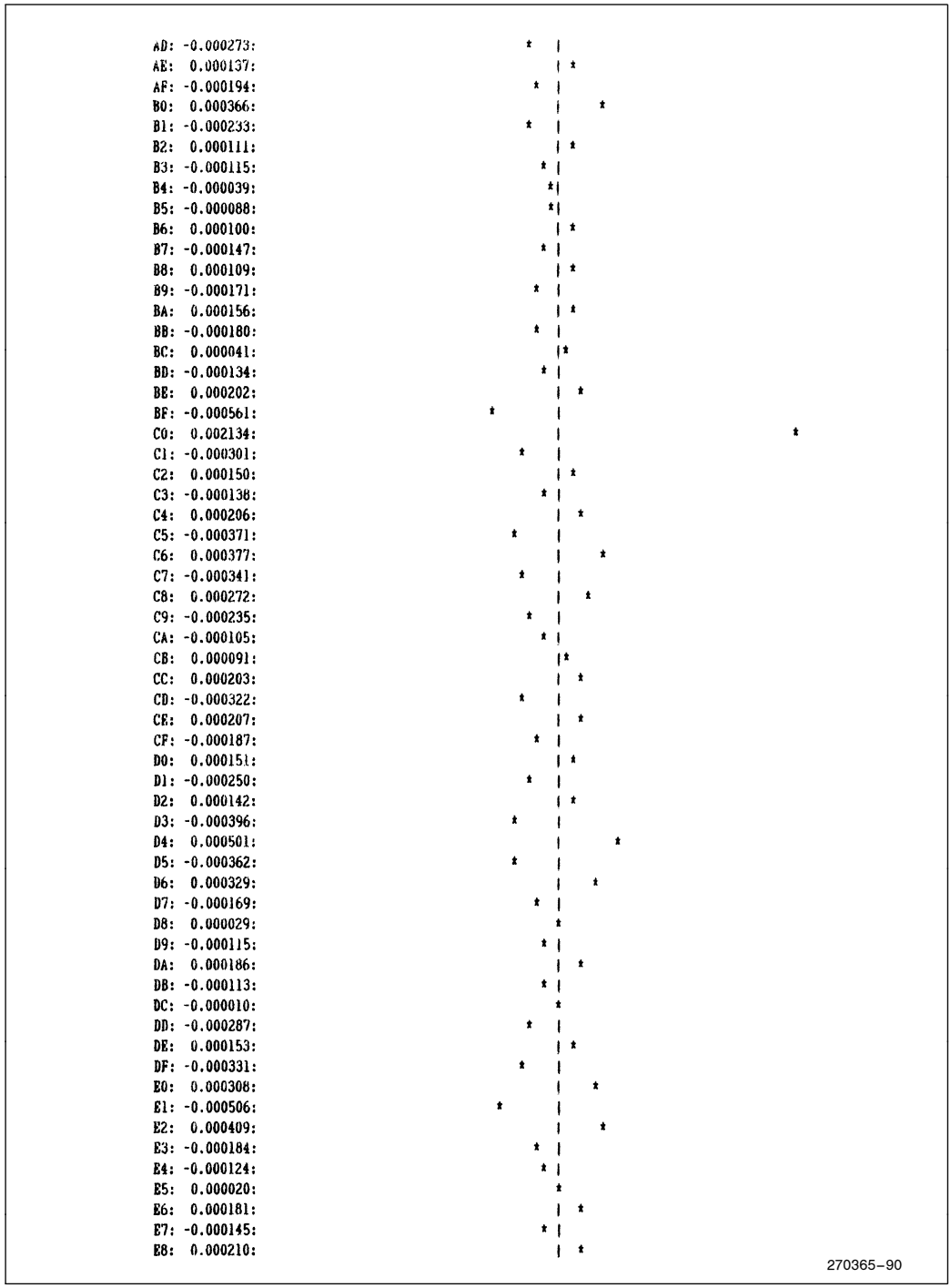
DNL Error, SN = 4130



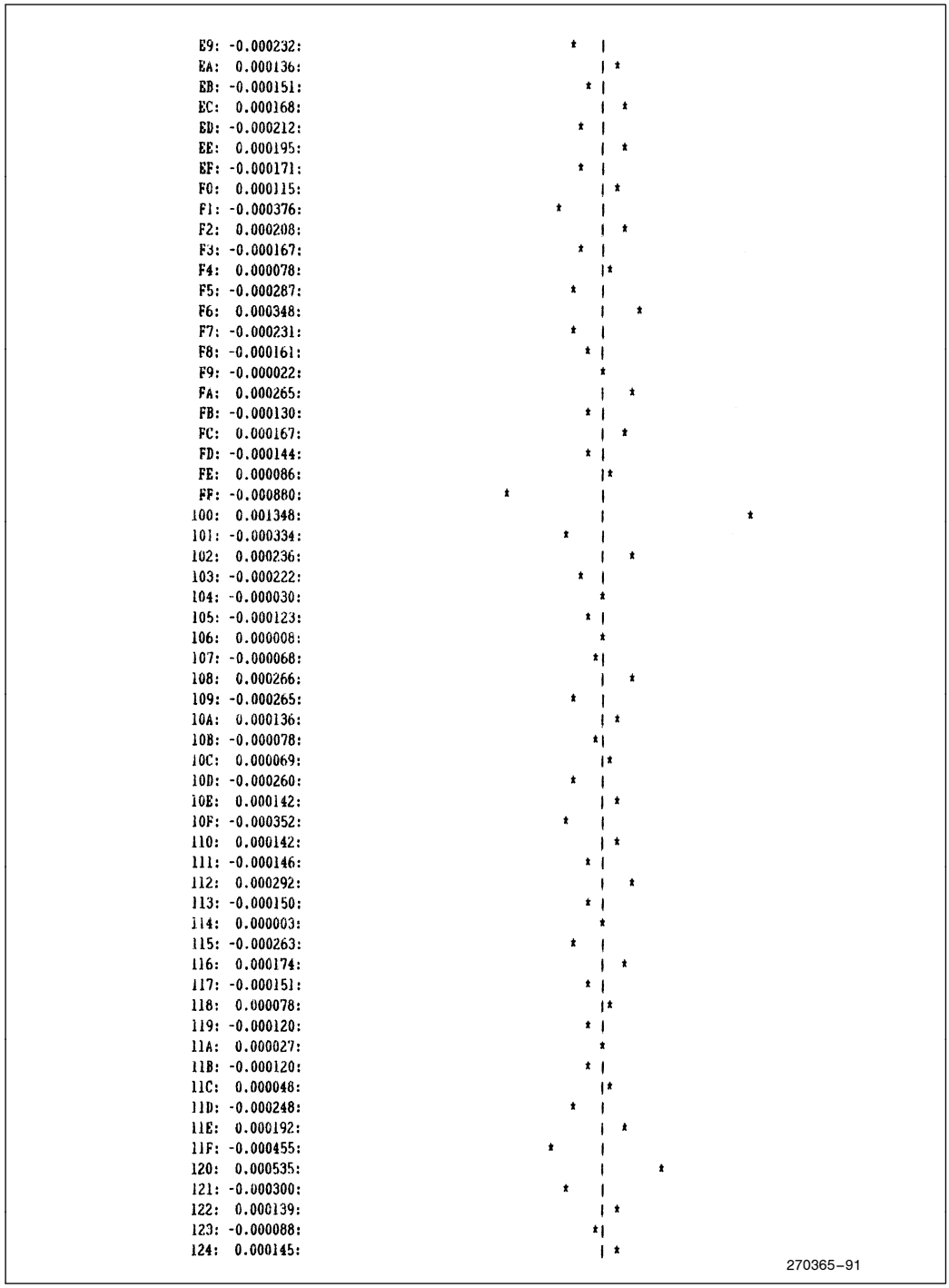
DNL Error, SN = 4130 (Continued)



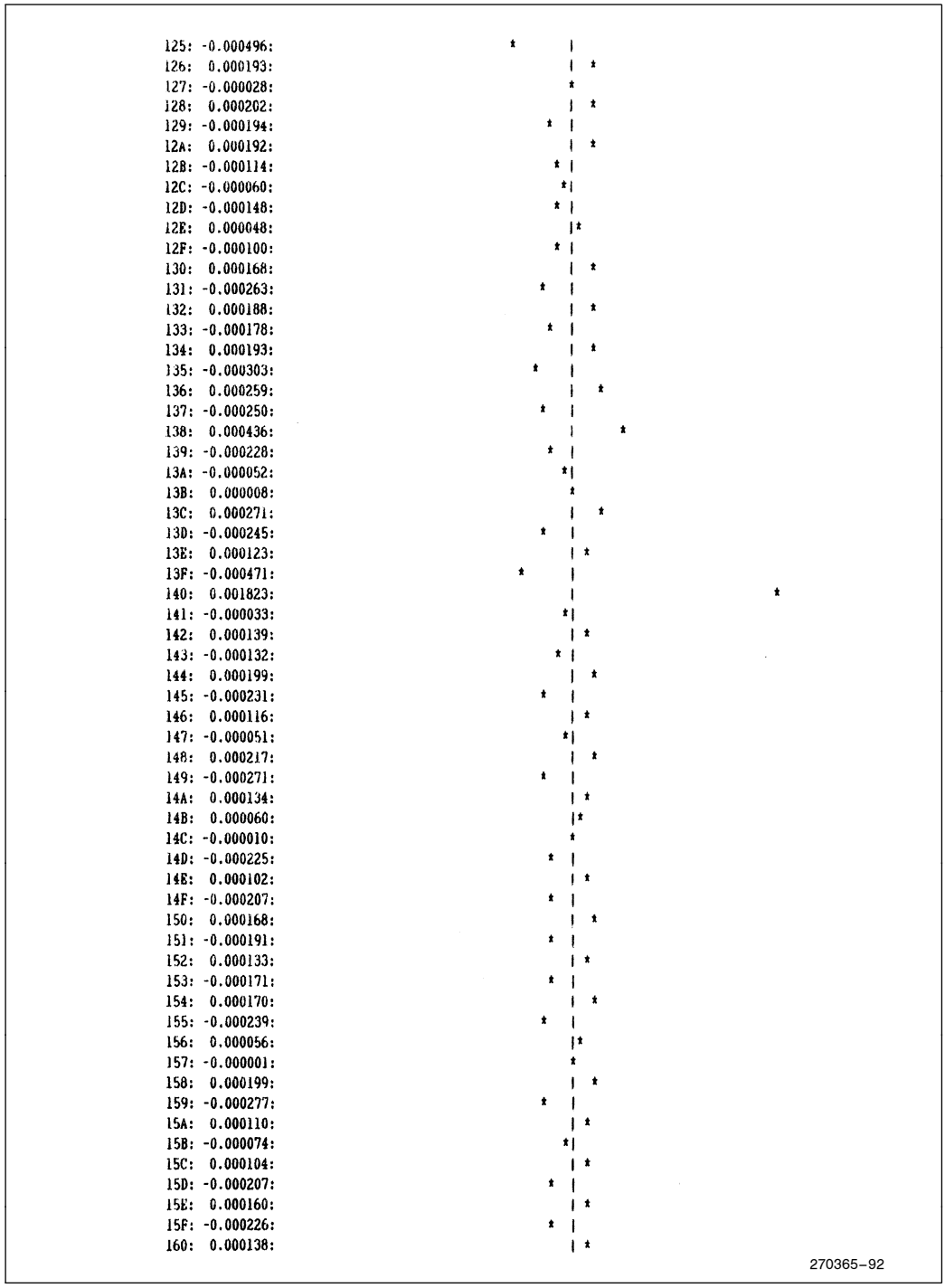
DNL Error, SN = 4130 (Continued)



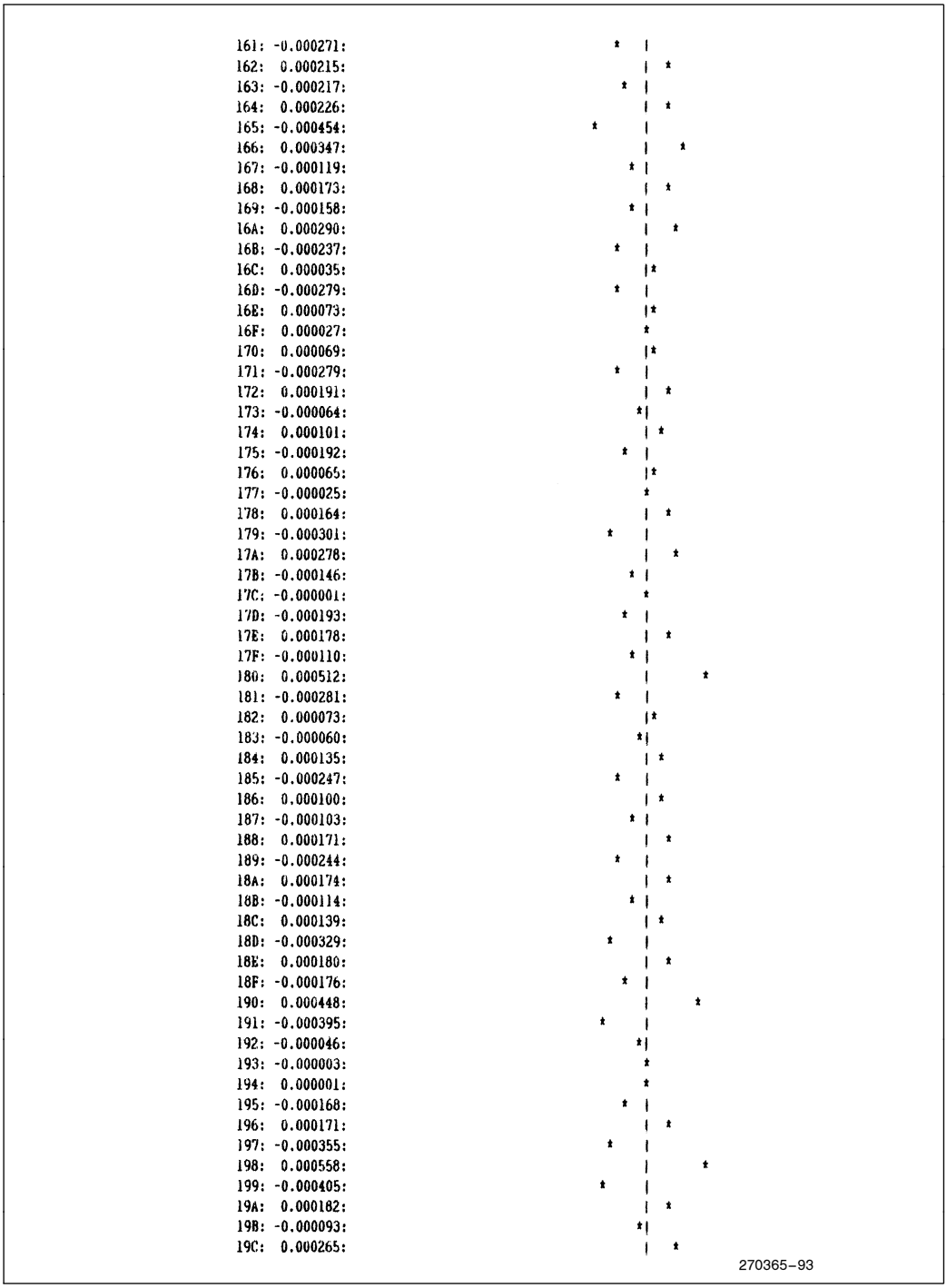
DNL Error, SN = 4130 (Continued)



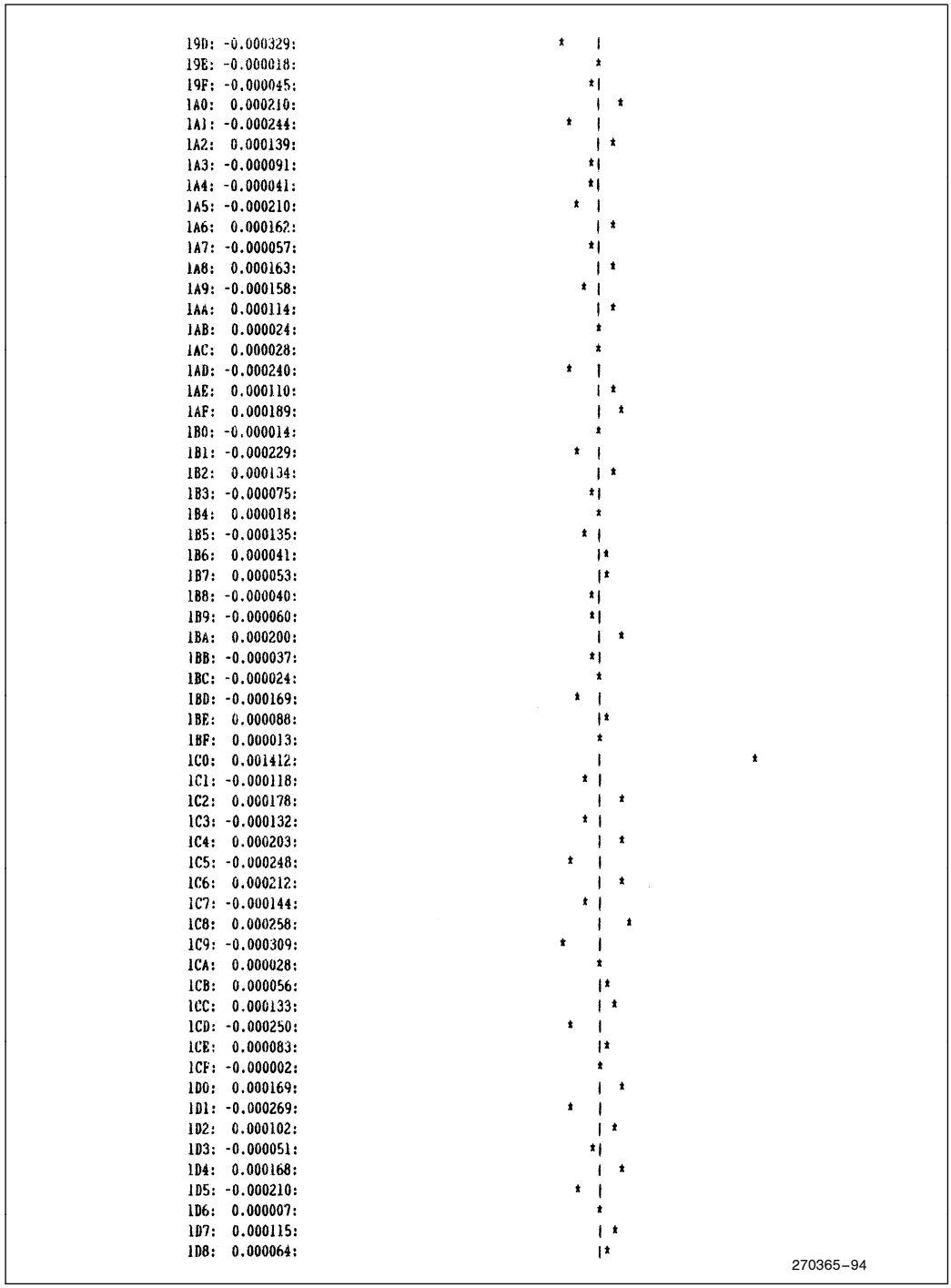
DNL Error, SN = 4130 (Continued)



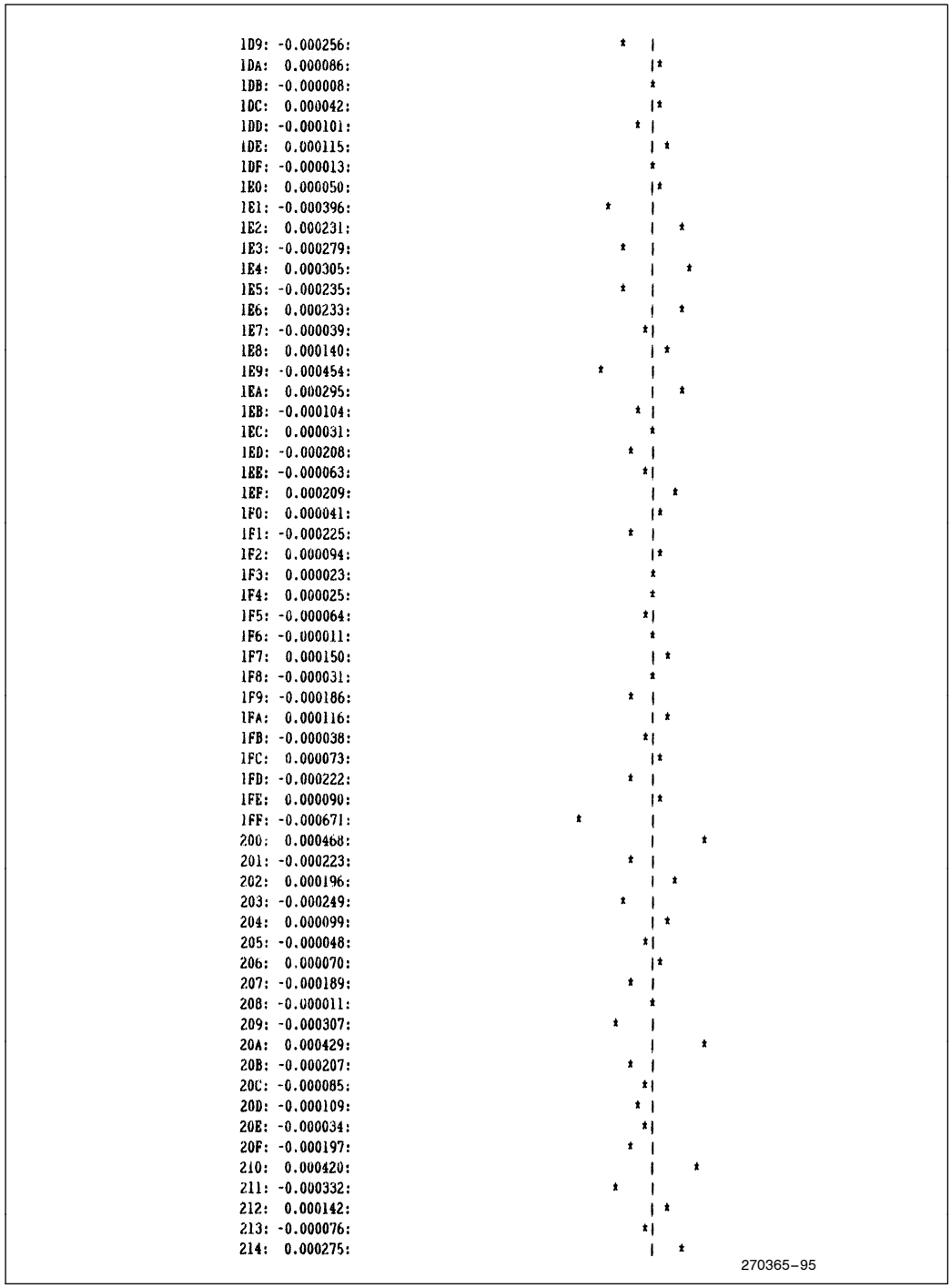
DNL Error, SN = 4130 (Continued)



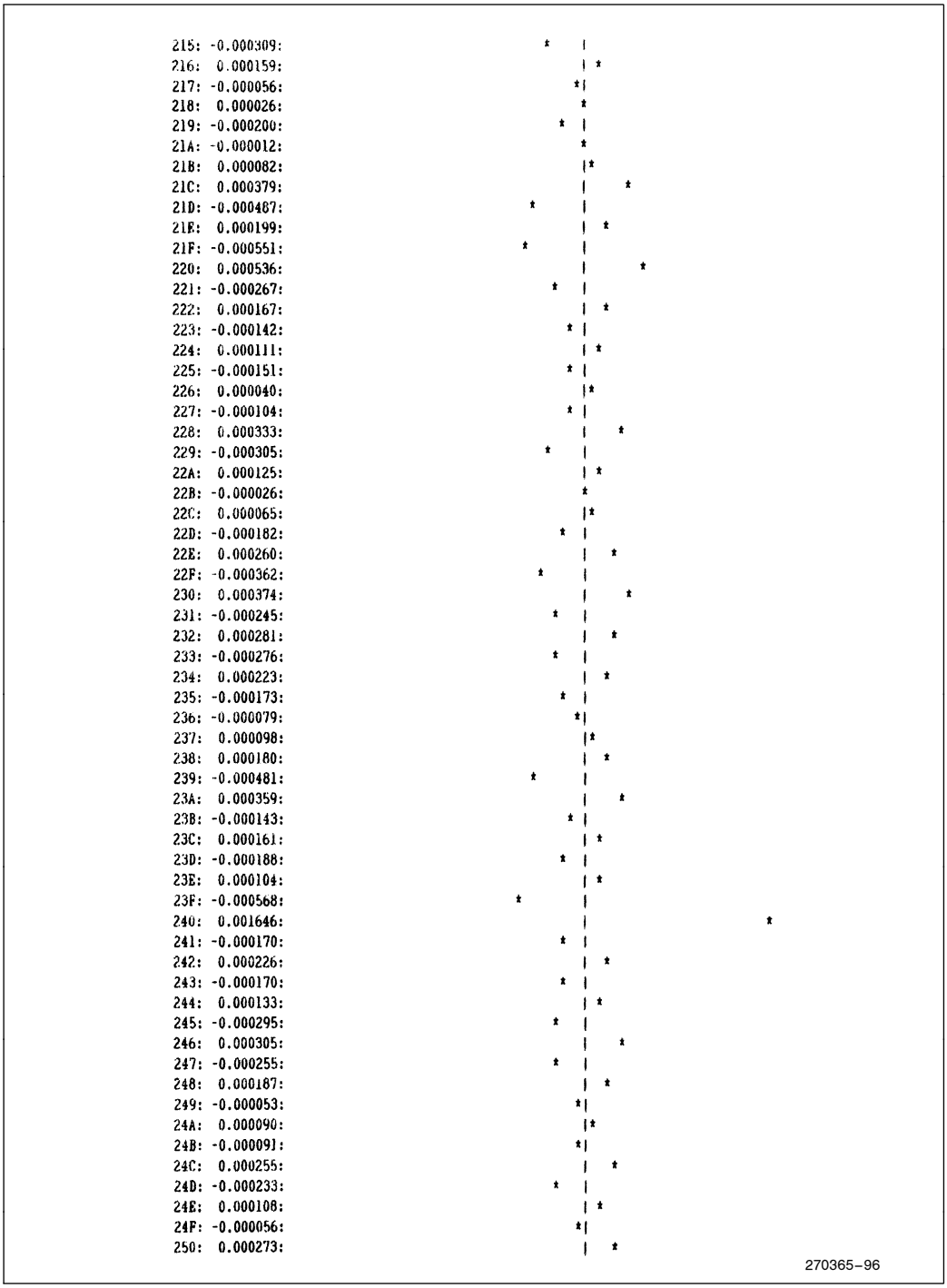
DNL Error, SN = 4130 (Continued)



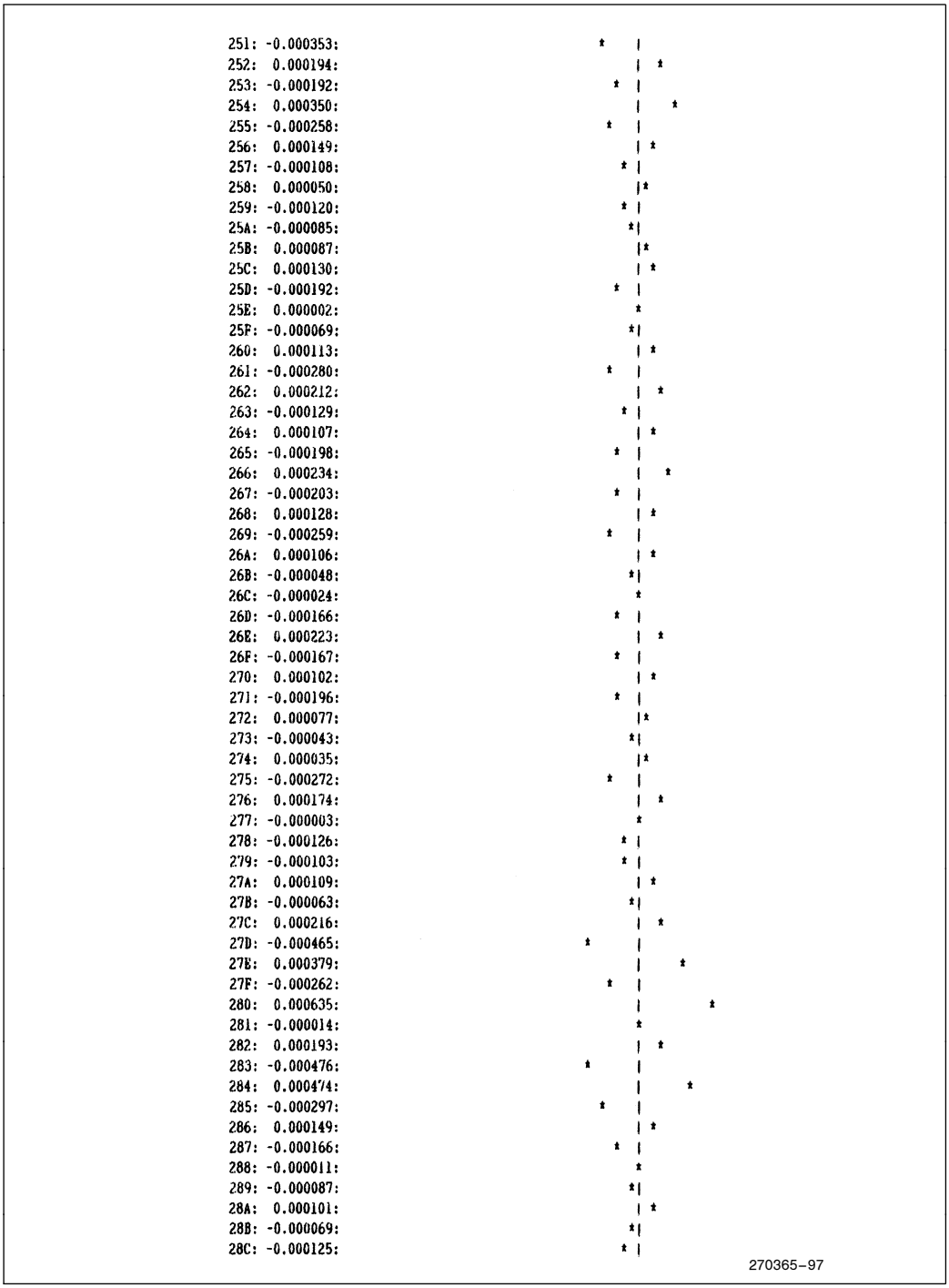
DNL Error, SN = 4130 (Continued)



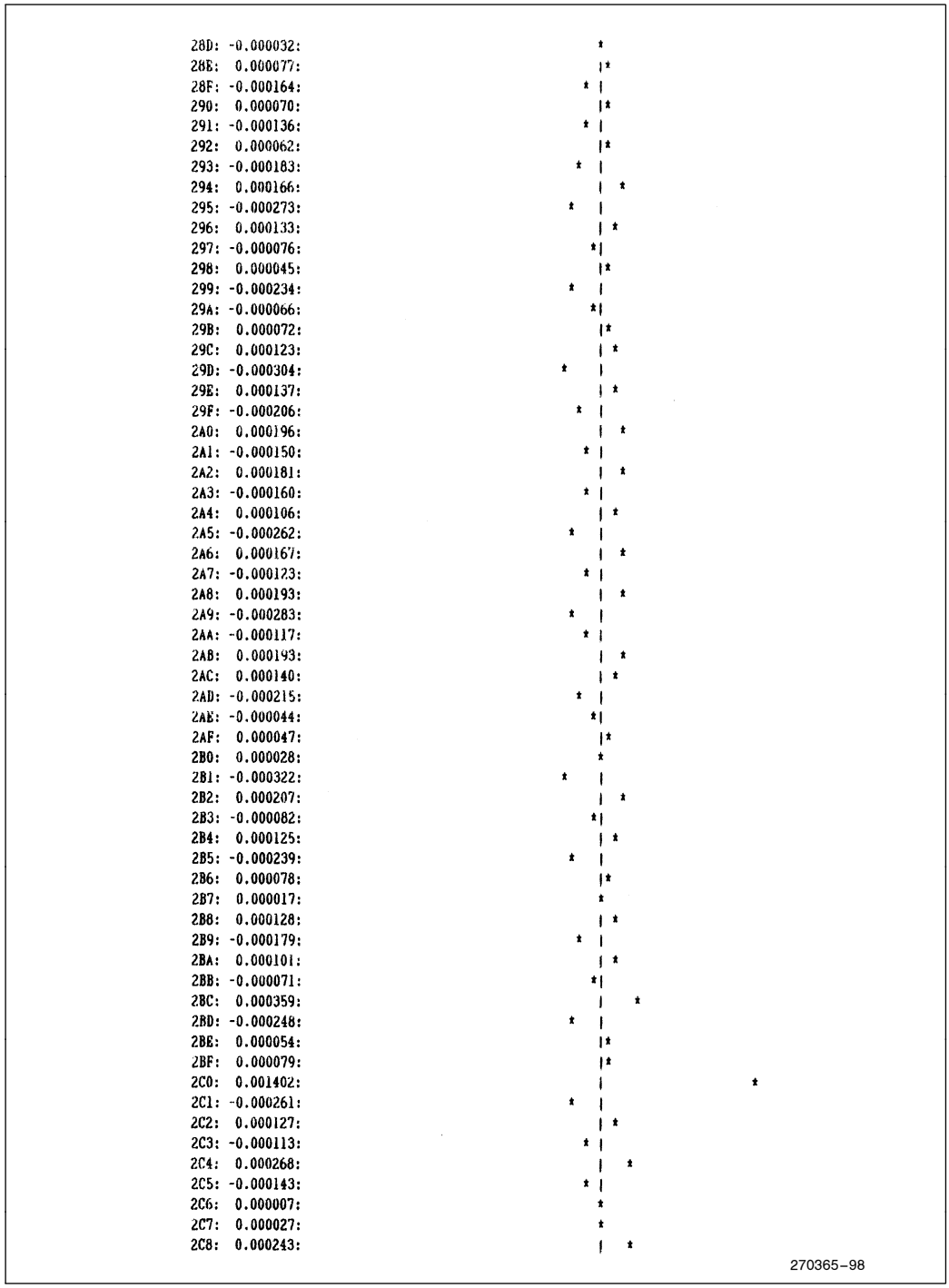
DNL Error, SN = 4130 (Continued)



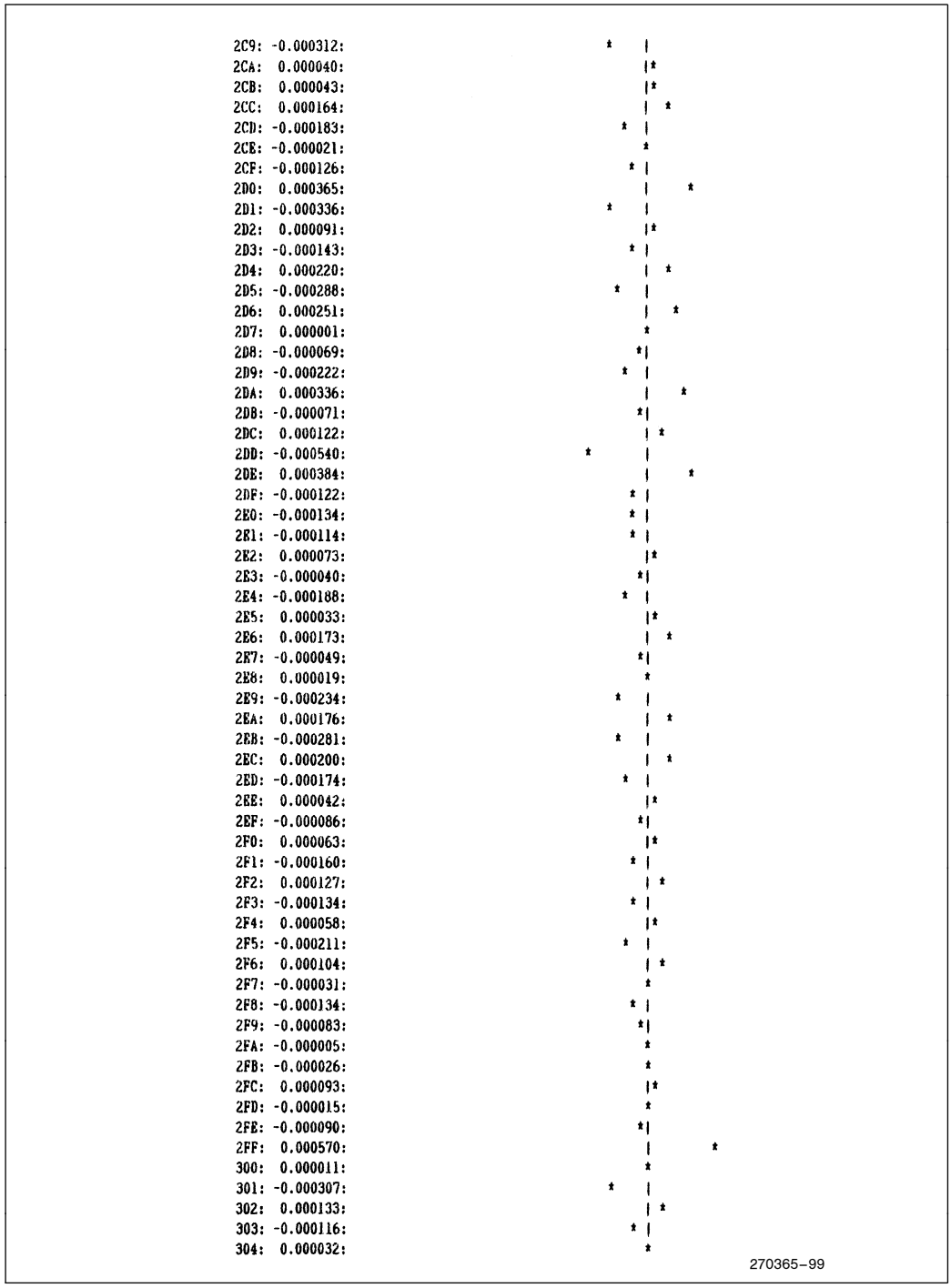
DNL Error, SN = 4130 (Continued)



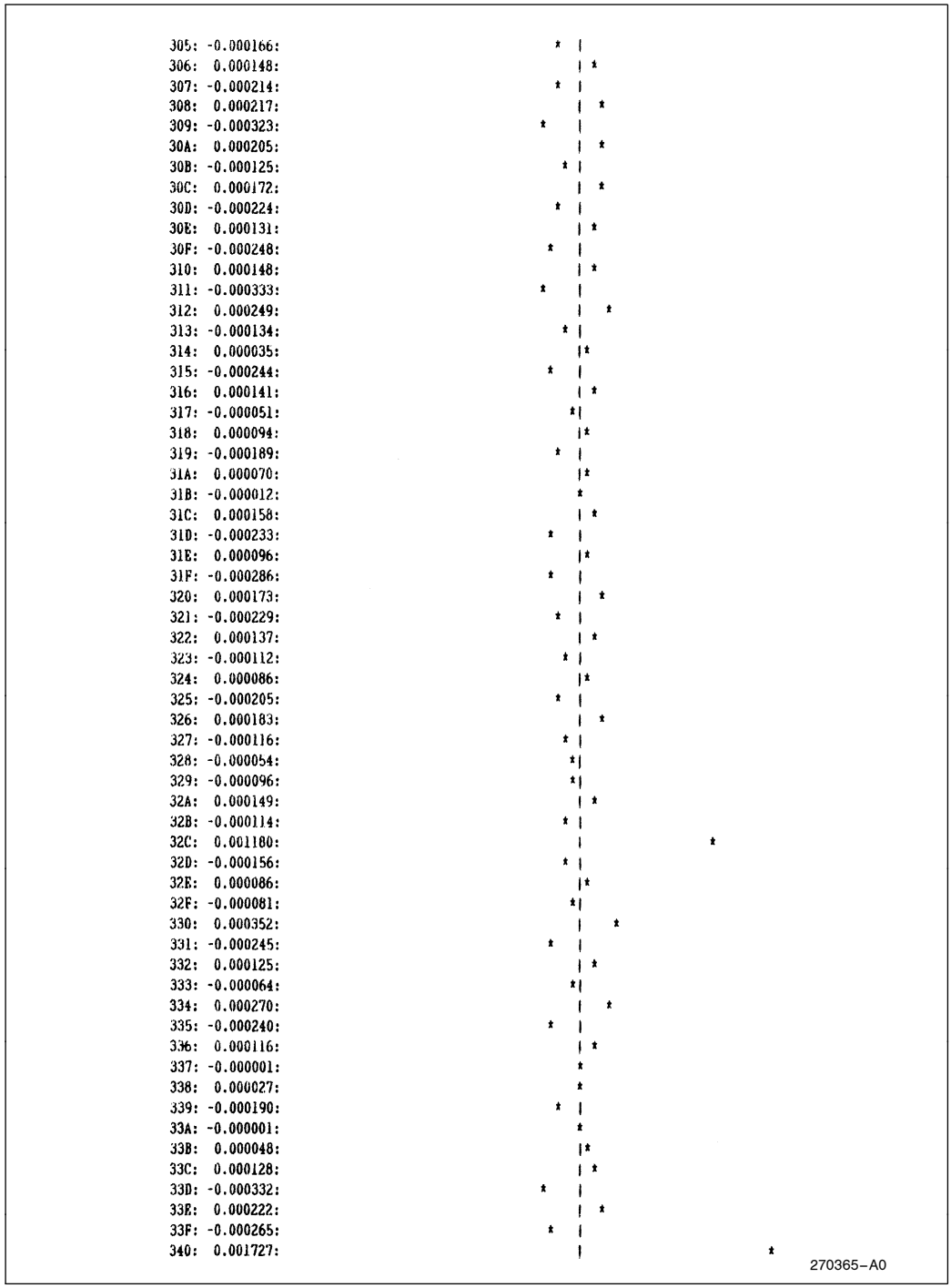
DNL Error, SN = 4130 (Continued)



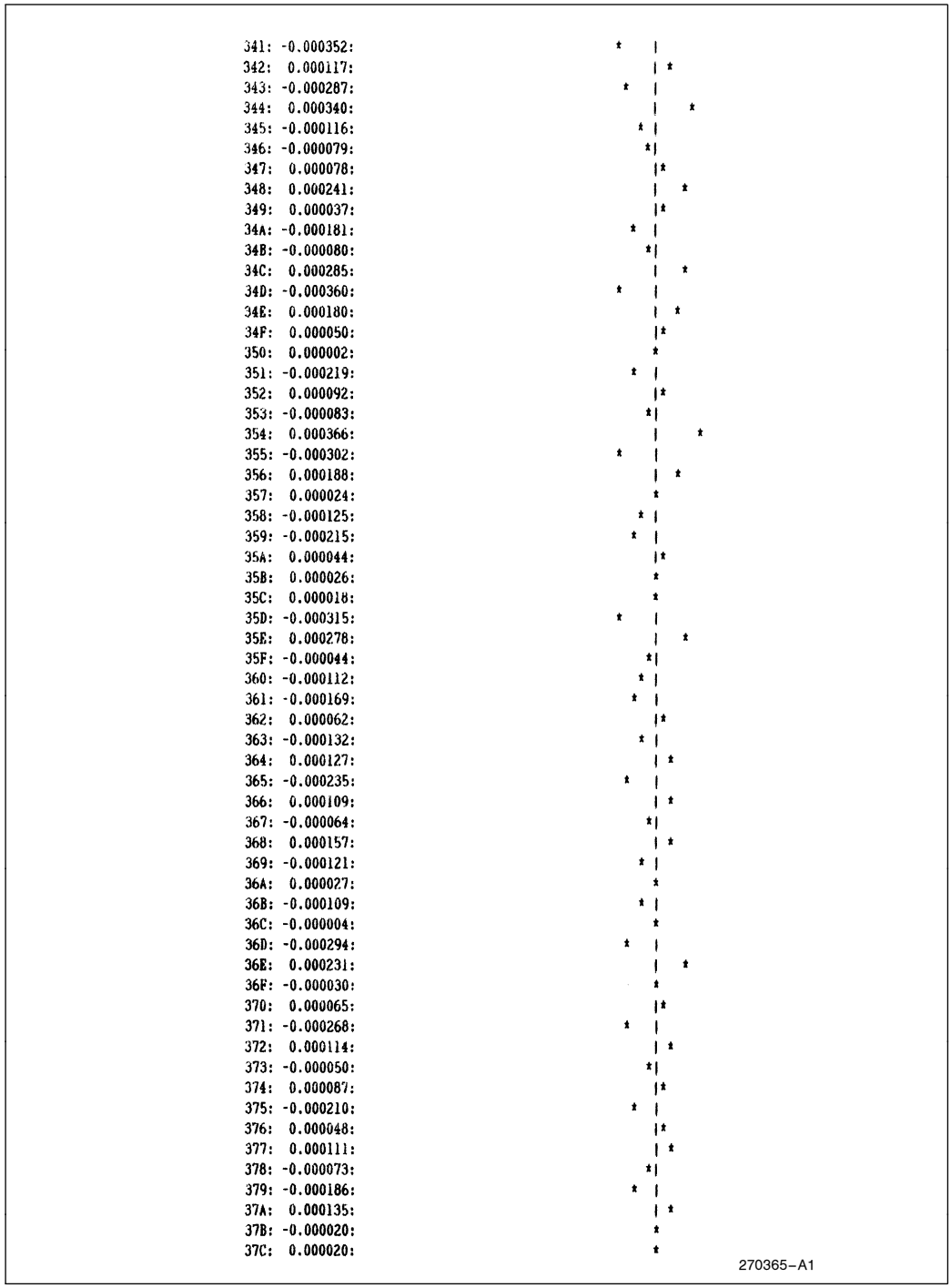
DNL Error, SN = 4130 (Continued)



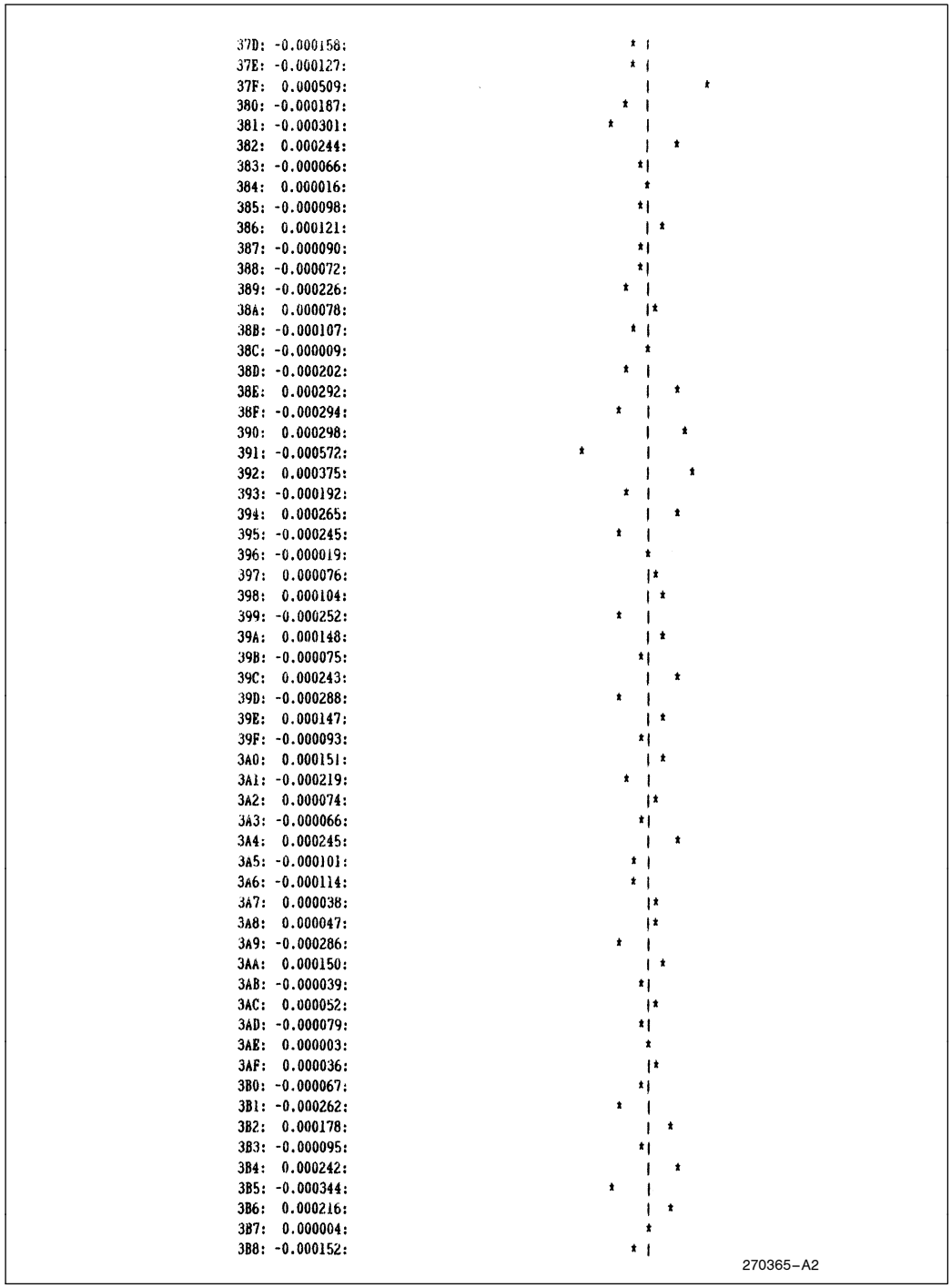
DNL Error, SN = 4130 (Continued)



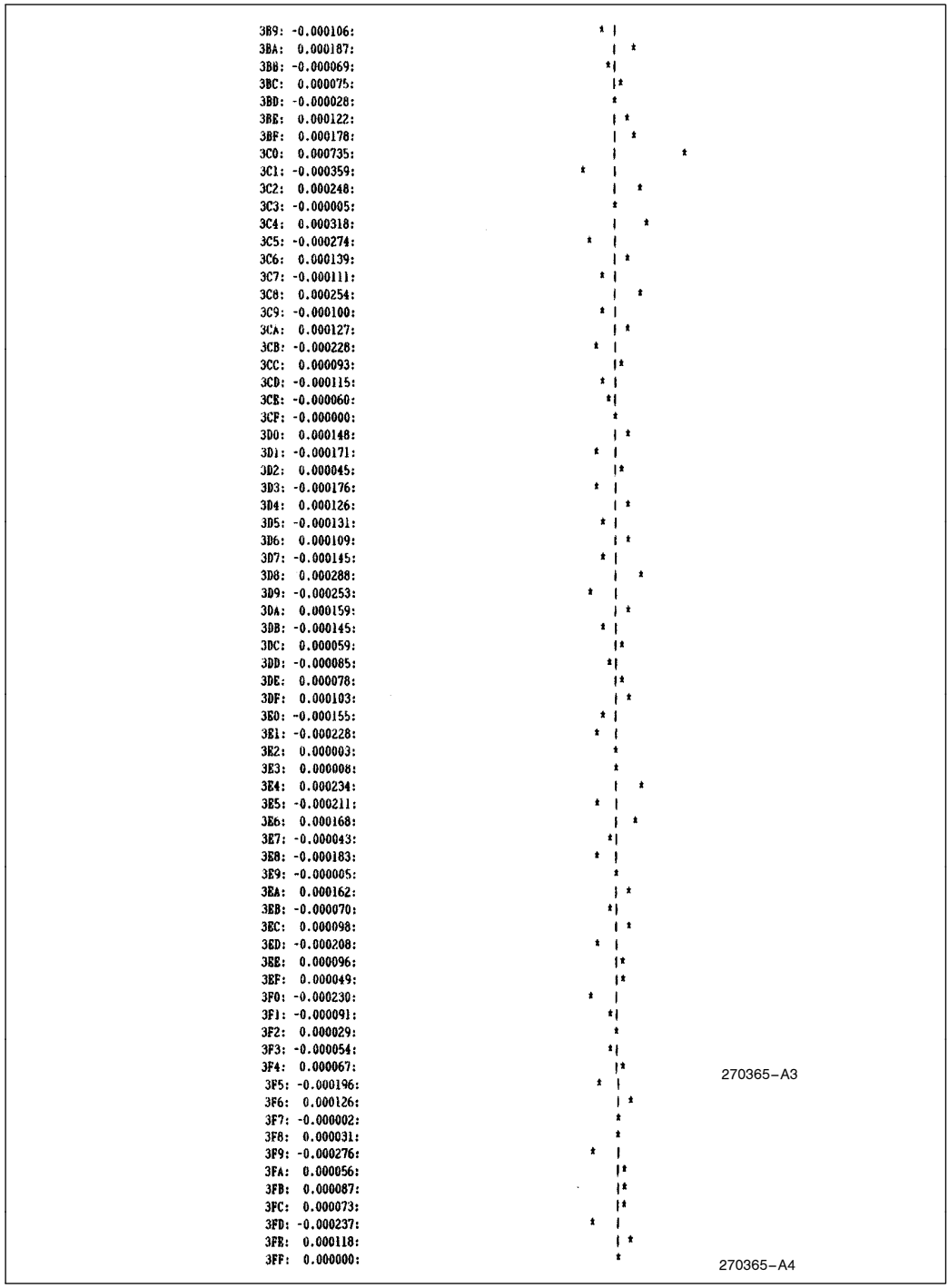
DNL Error, SN = 4130 (Continued)



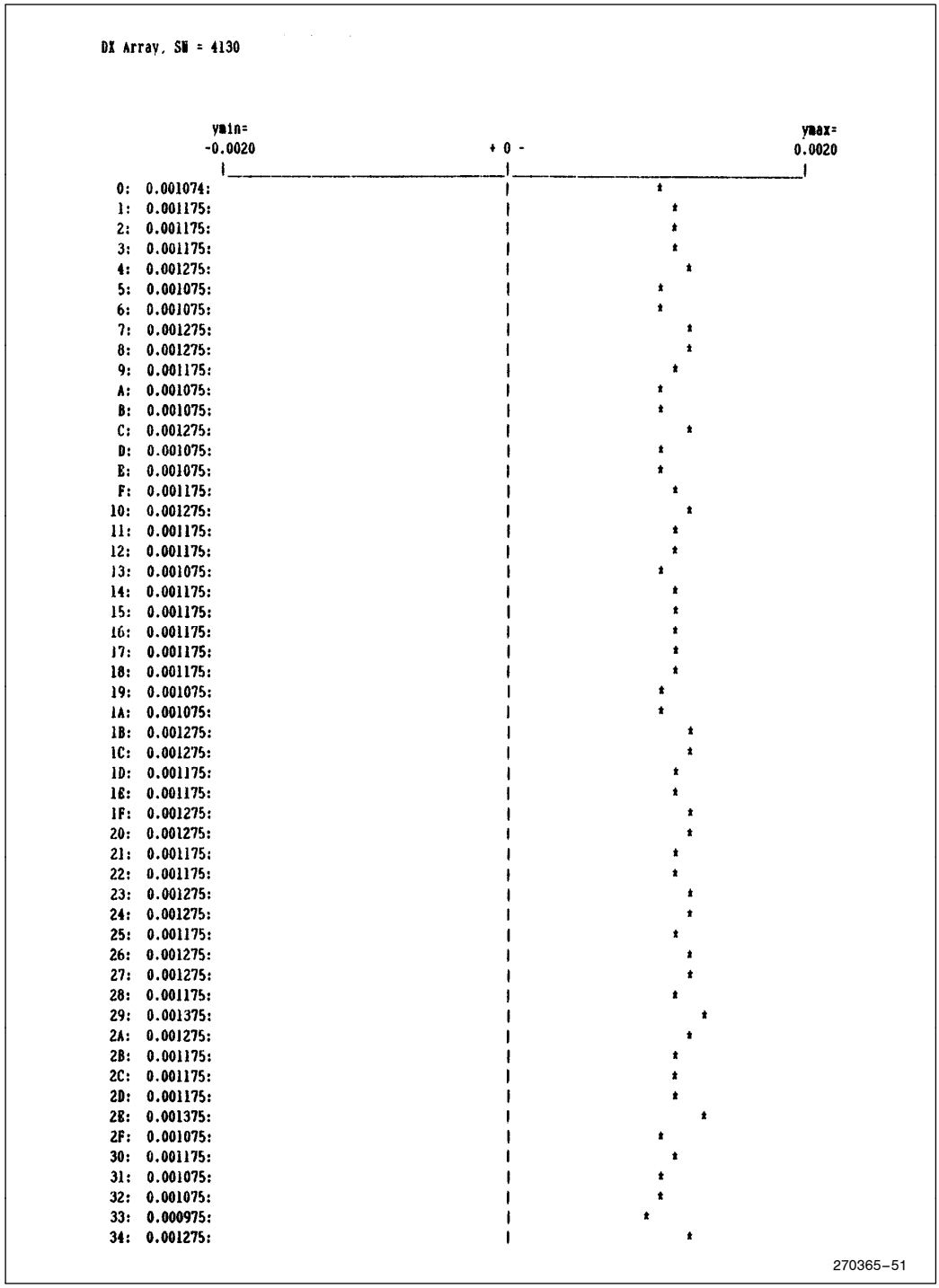
DNL Error, SN = 4130 (Continued)



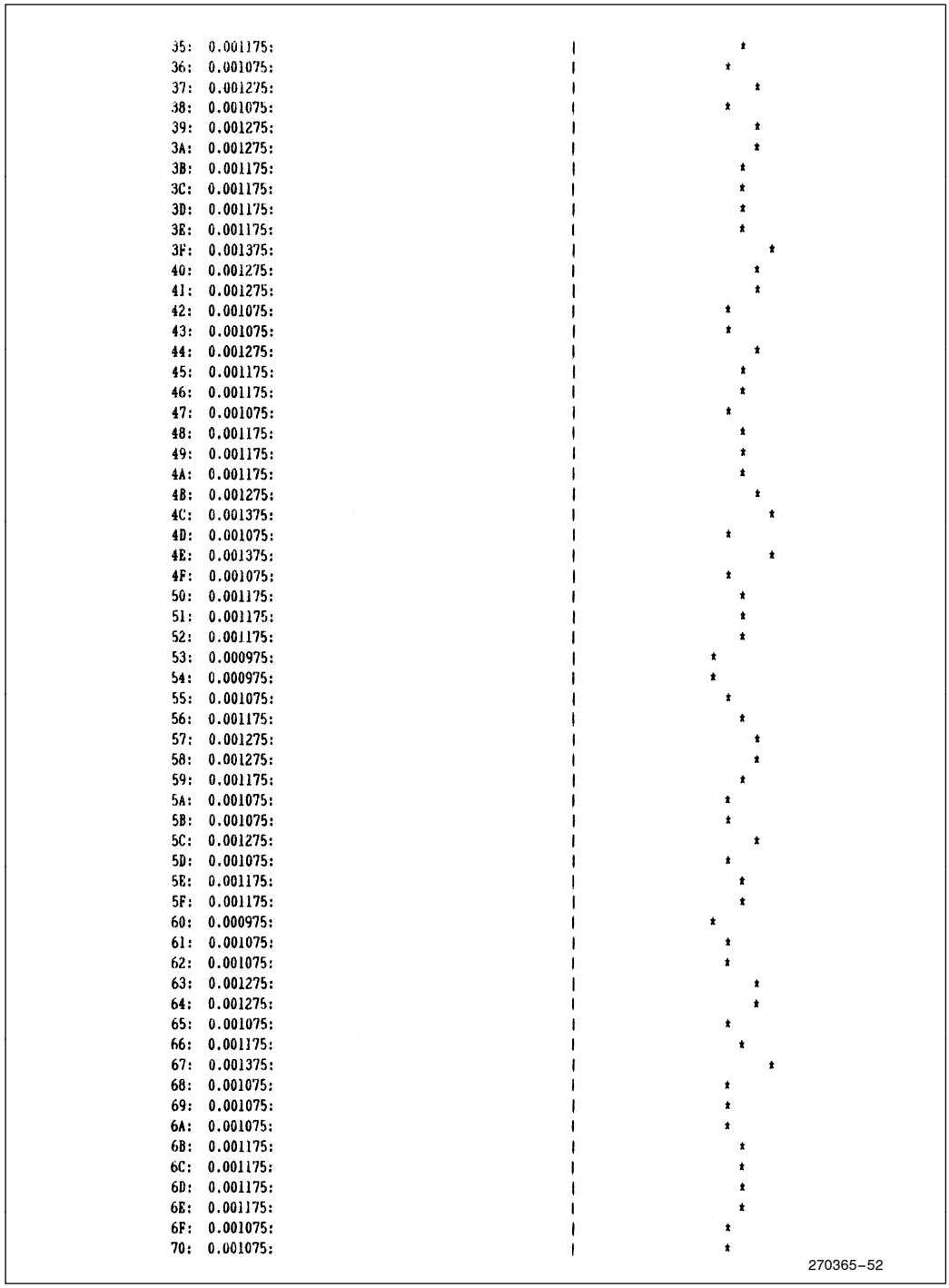
DNL Error, SN = 4130 (Continued)



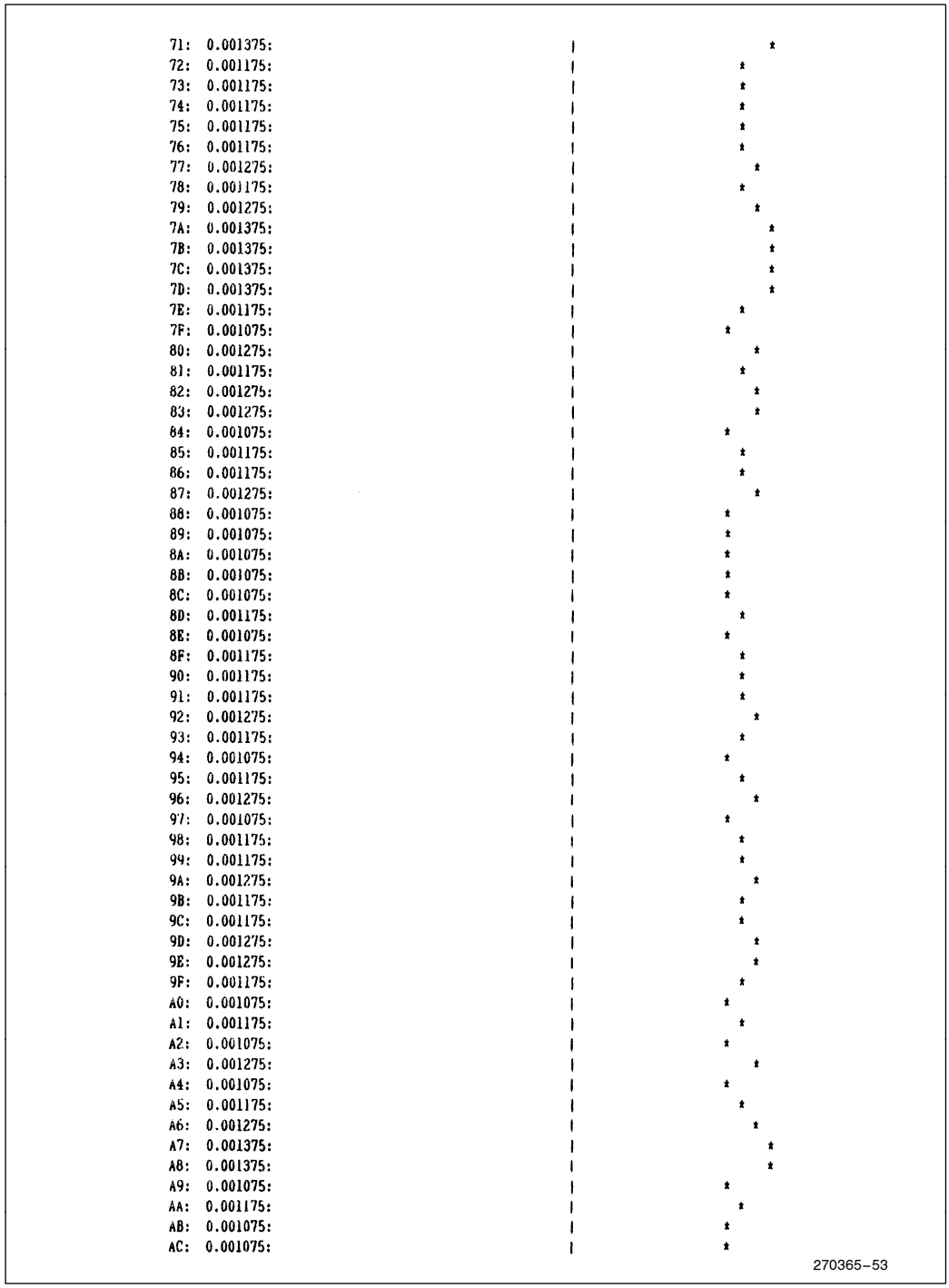
DNL Error, SN = 4130 (Continued)



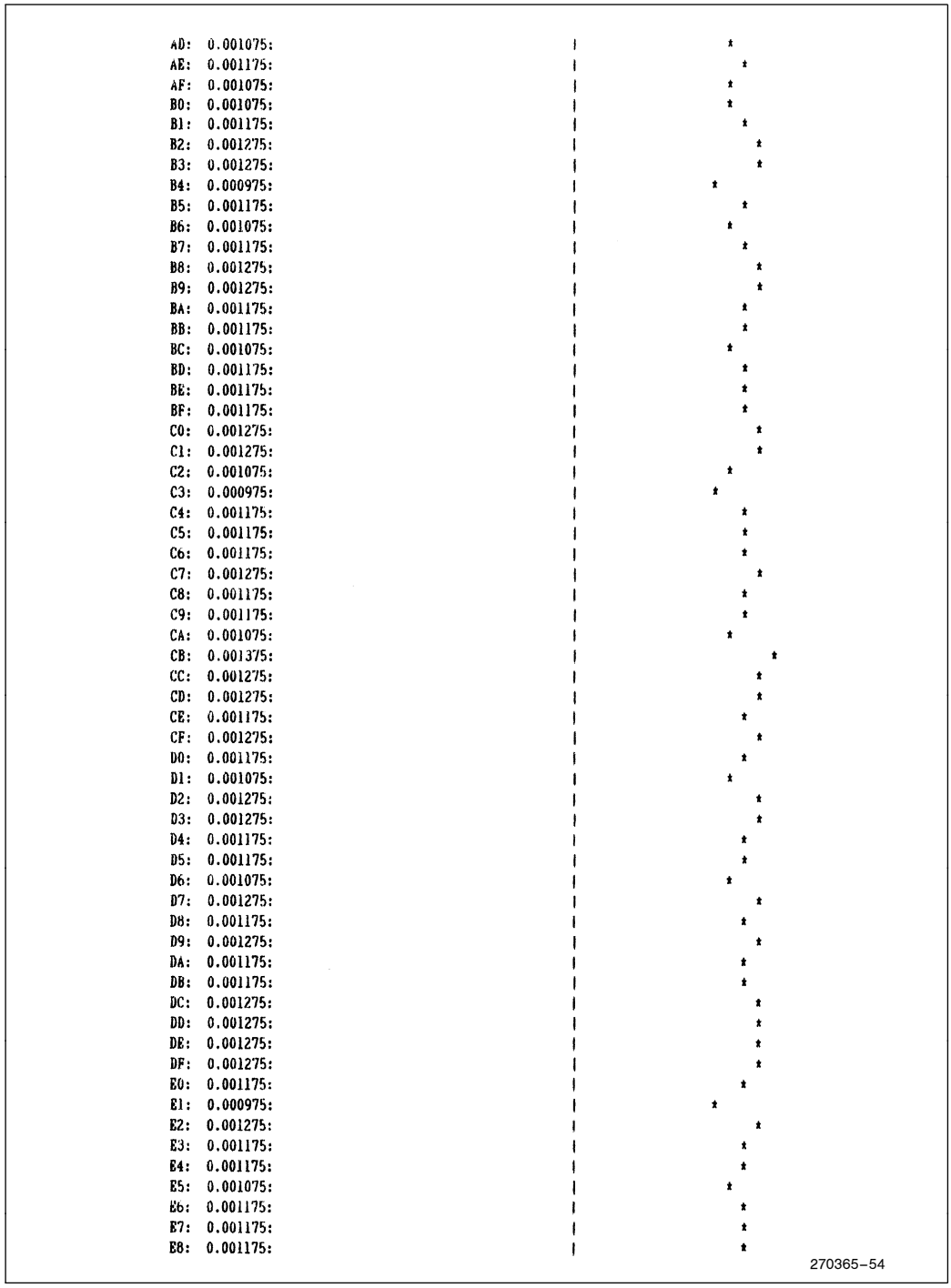
Repeatability Error, SN = 4130



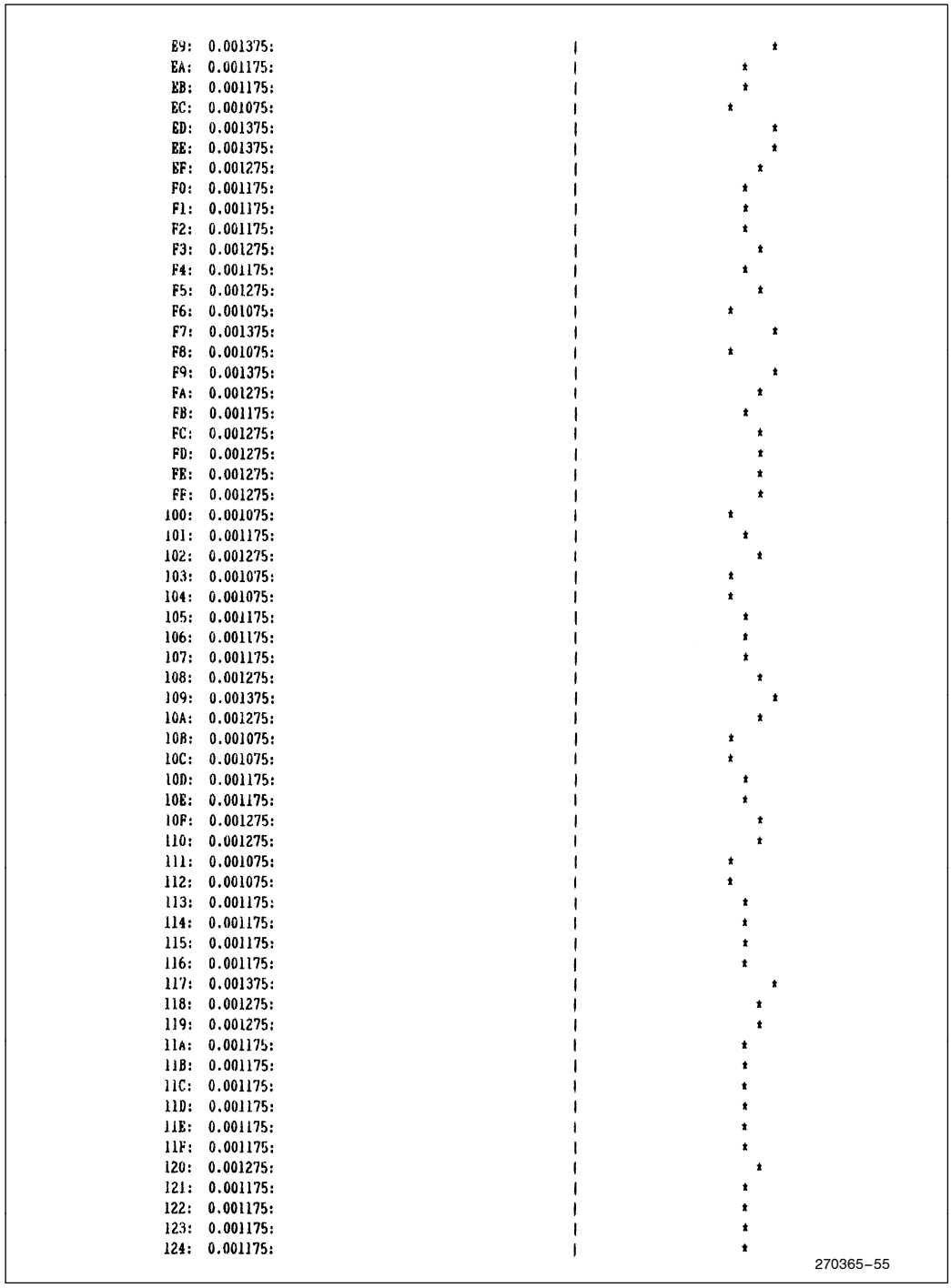
Repeatability Error, SN = 4130 (Continued)



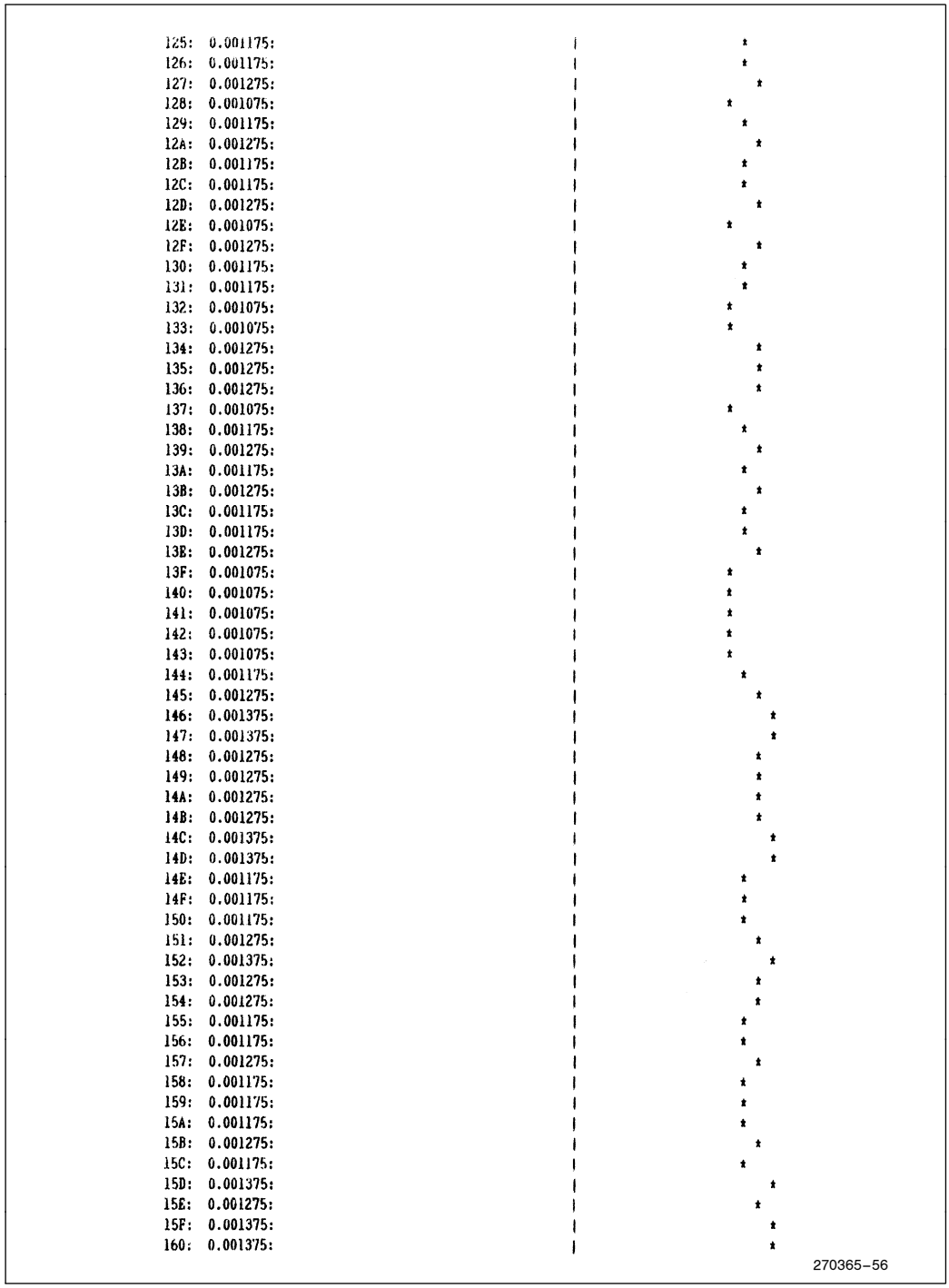
Repeatability Error, SN = 4130 (Continued)



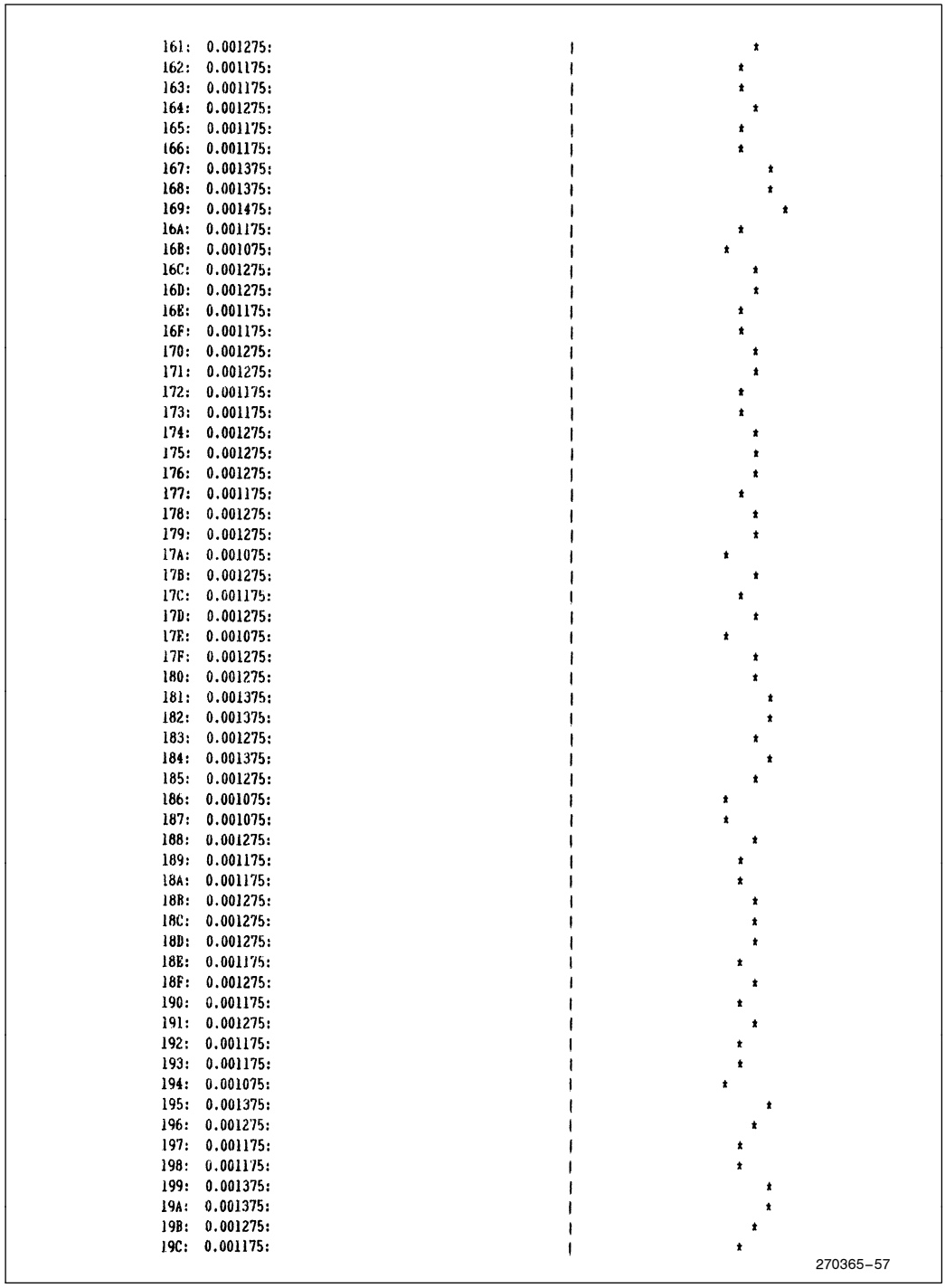
Repeatability Error, SN = 4130 (Continued)



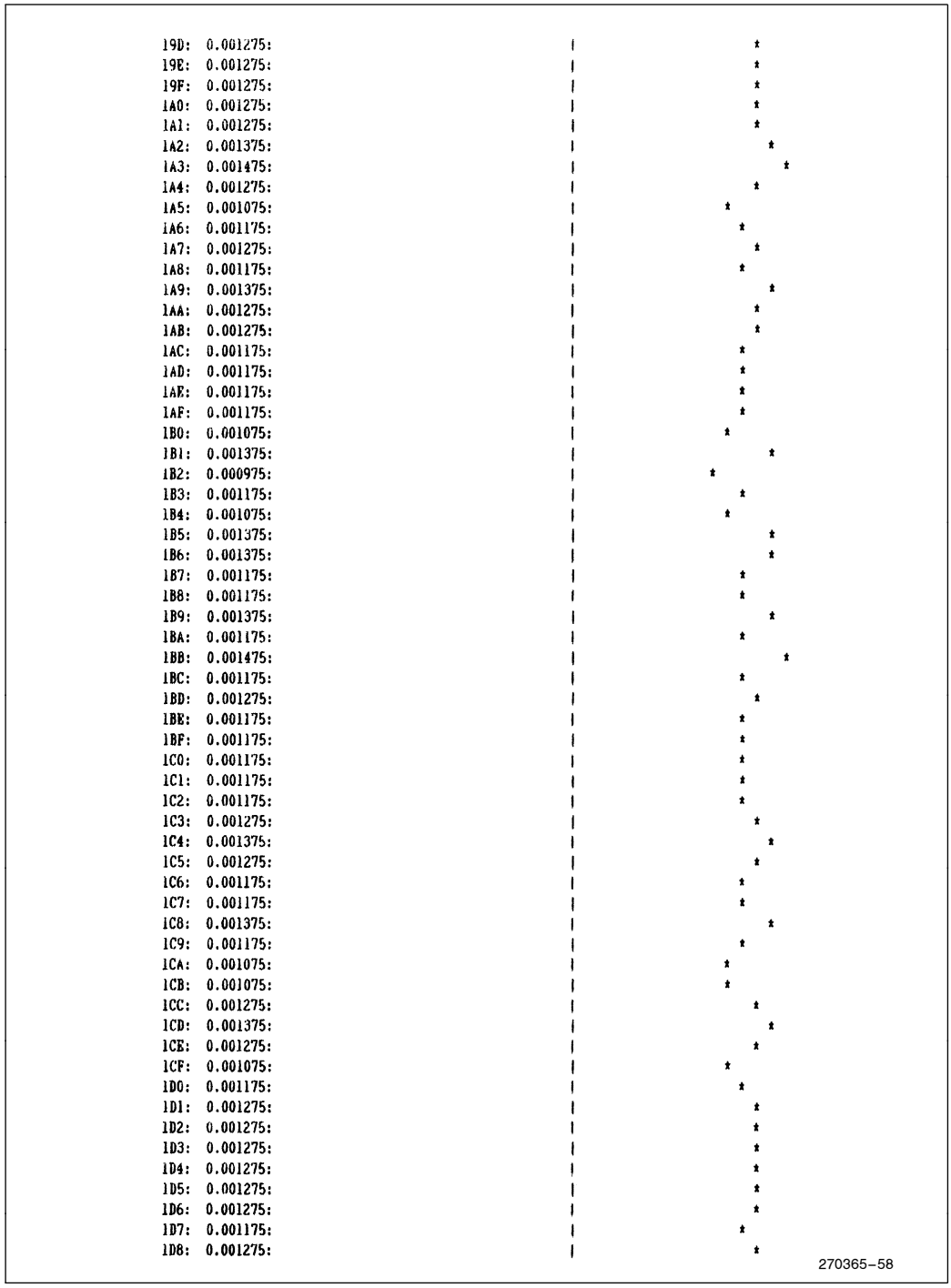
Repeatability Error, SN = 4130 (Continued)



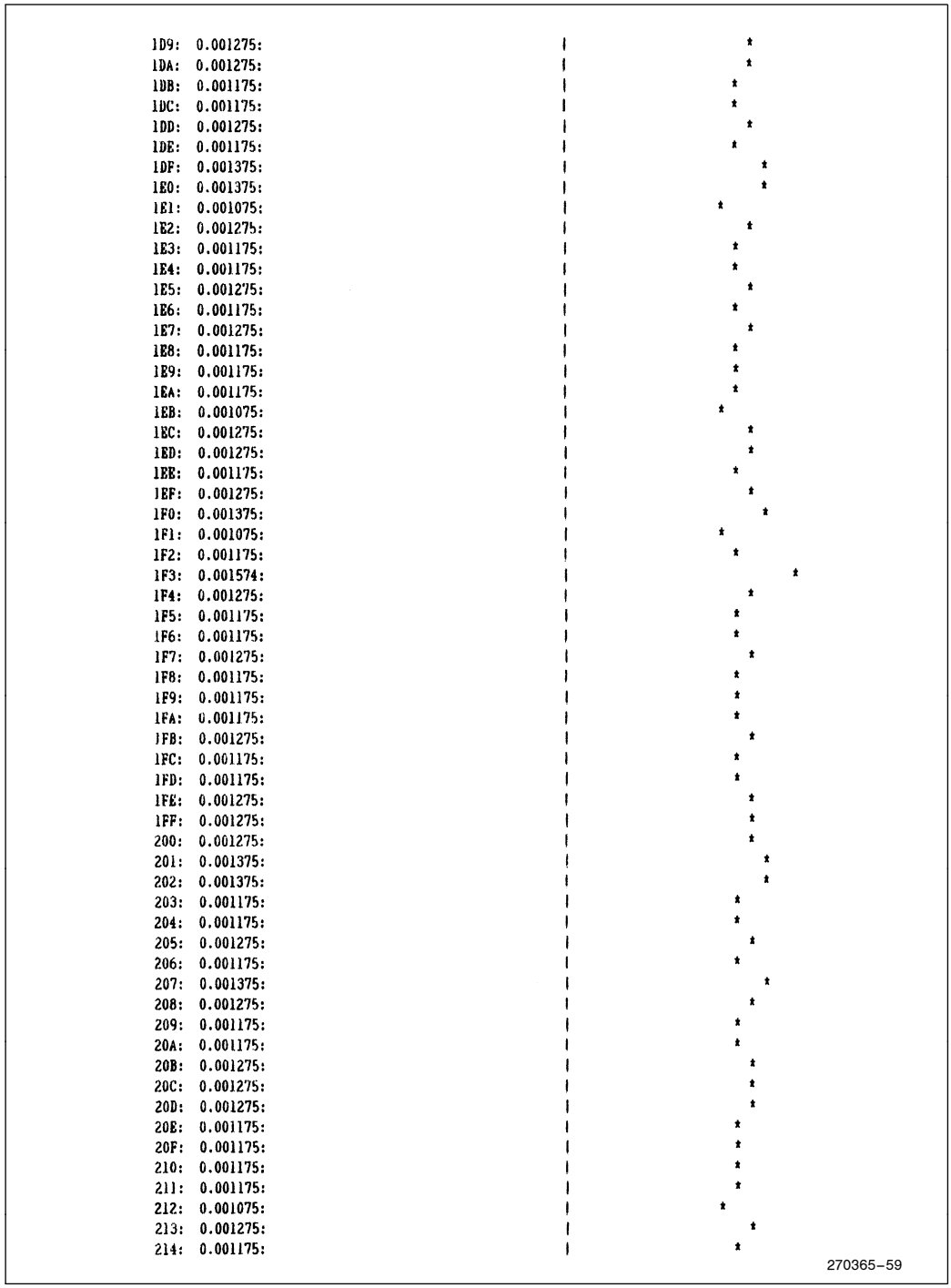
Repeatability Error, SN = 4130 (Continued)



Repeatability Error, SN = 4130 (Continued)



Repeatability Error, SN = 4130 (Continued)

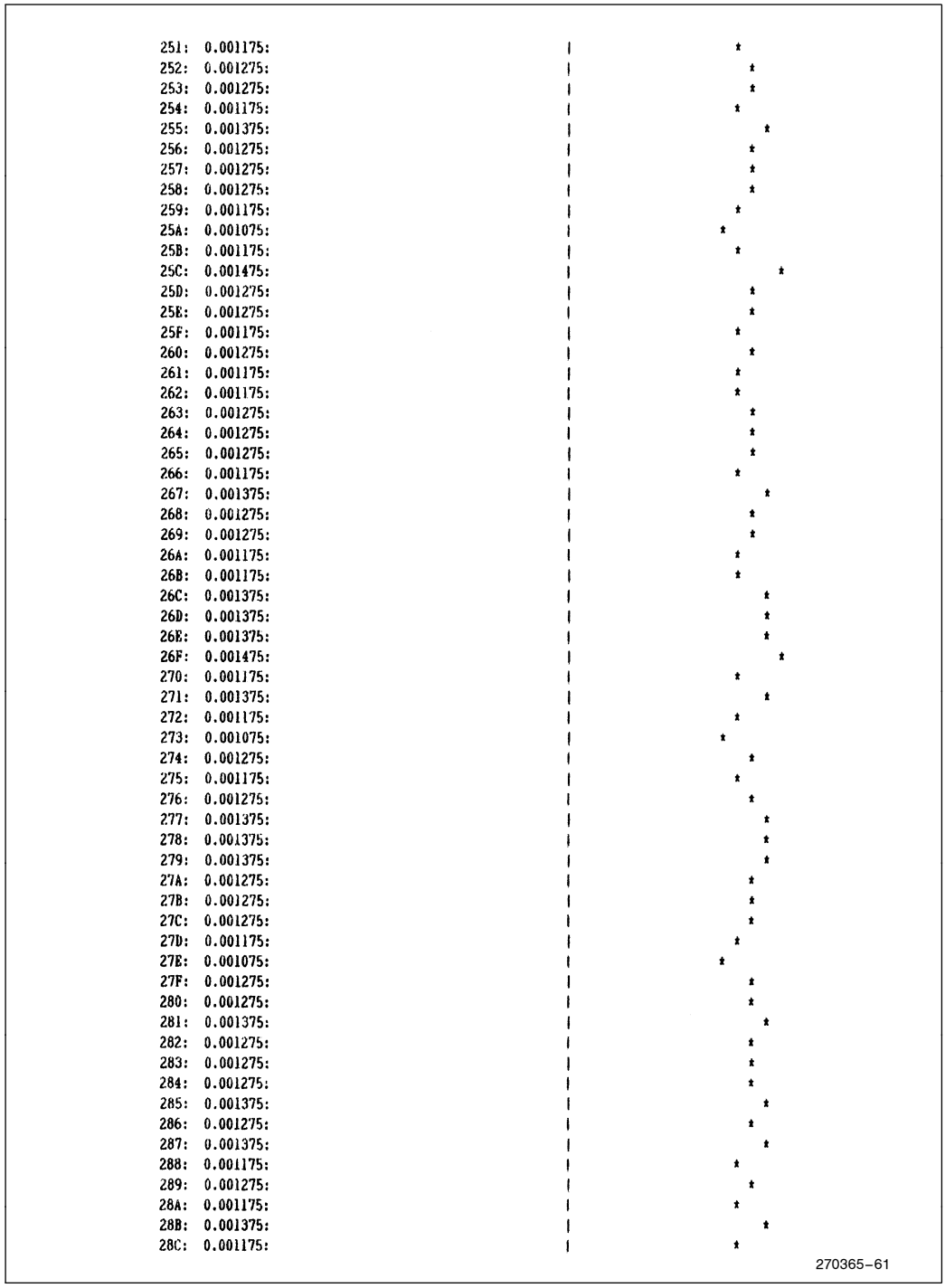


Repeatability Error, SN = 4130 (Continued)

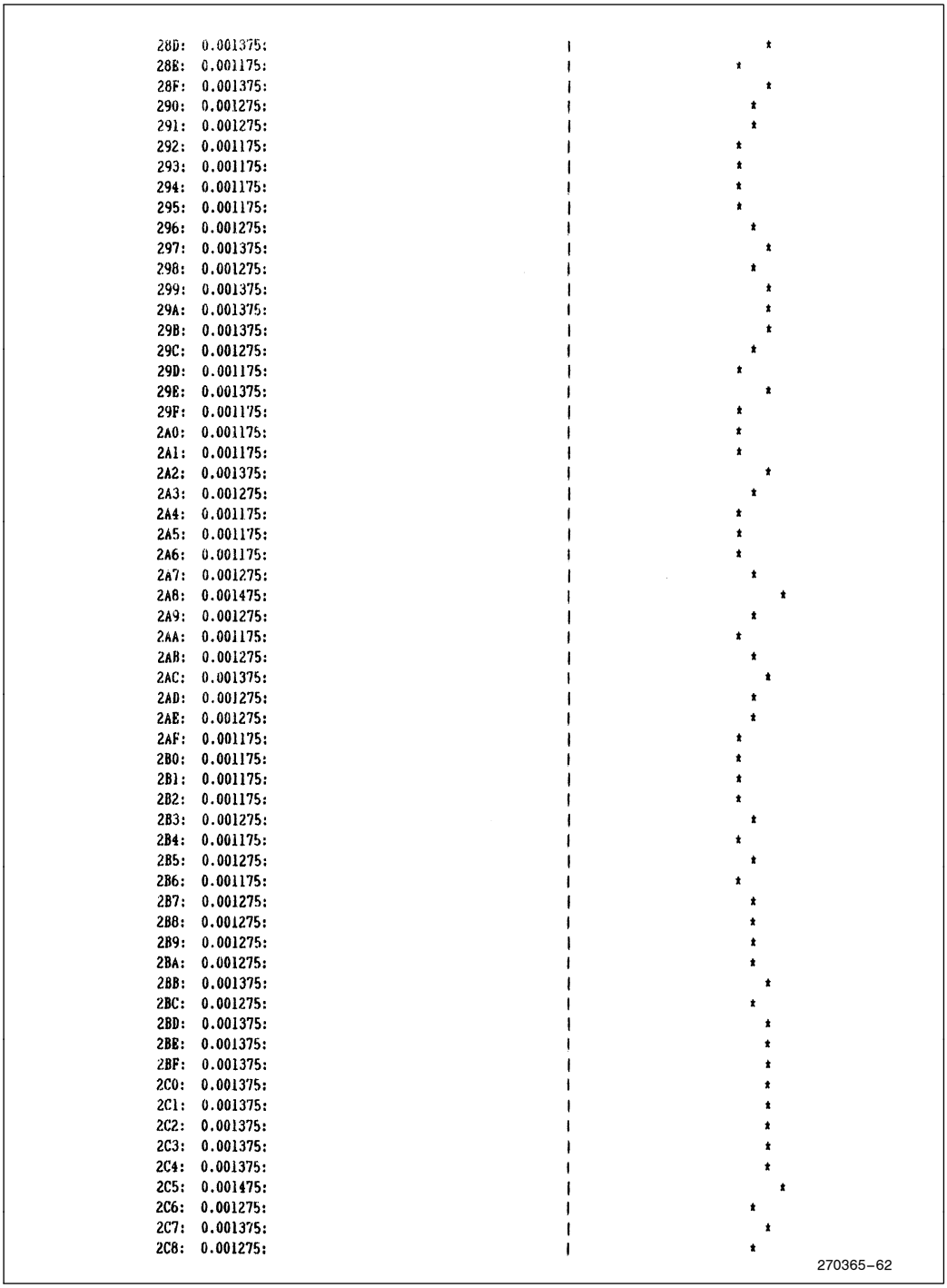
```
215:  0.001275:                                                  *
216:  0.001275:                                                  *
217:  0.001275:                                                  *
218:  0.001275:                                                  *
219:  0.001375:                                                    *
21A:  0.001175:                                                *
21B:  0.001275:                                                  *
21C:  0.001275:                                                  *
21D:  0.001275:                                                  *
21E:  0.001175:                                                *
21F:  0.001175:                                                *
220:  0.001375:                                                    *
221:  0.001275:                                                  *
222:  0.001375:                                                    *
223:  0.001375:                                                    *
224:  0.001175:                                                *
225:  0.001375:                                                    *
226:  0.001175:                                                *
227:  0.001375:                                                    *
228:  0.001175:                                                *
229:  0.001275:                                                  *
22A:  0.001175:                                                *
22B:  0.001275:                                                  *
22C:  0.001275:                                                  *
22D:  0.001175:                                                *
22E:  0.001175:                                                *
22F:  0.001075:                                               *
230:  0.001275:                                                  *
231:  0.001175:                                                *
232:  0.001175:                                                *
233:  0.001275:                                                  *
234:  0.001275:                                                  *
235:  0.001375:                                                    *
236:  0.001375:                                                    *
237:  0.001275:                                                  *
238:  0.001275:                                                  *
239:  0.001175:                                                *
23A:  0.001175:                                                *
23B:  0.001175:                                                *
23C:  0.001175:                                                *
23D:  0.001375:                                                    *
23E:  0.001175:                                                *
23F:  0.001275:                                                  *
240:  0.001275:                                                  *
241:  0.001275:                                                  *
242:  0.001275:                                                  *
243:  0.001275:                                                  *
244:  0.001275:                                                  *
245:  0.001375:                                                    *
246:  0.001075:                                               *
247:  0.001175:                                                *
248:  0.001175:                                                *
249:  0.001375:                                                    *
24A:  0.001175:                                                *
24B:  0.001275:                                                  *
24C:  0.001075:                                               *
24D:  0.001175:                                                *
24E:  0.001375:                                                    *
24F:  0.001375:                                                    *
250:  0.001275:                                                  *
```

270365–60

Repeatability Error, SN = 4130 (Continued)



Repeatability Error, SN = 4130 (Continued)

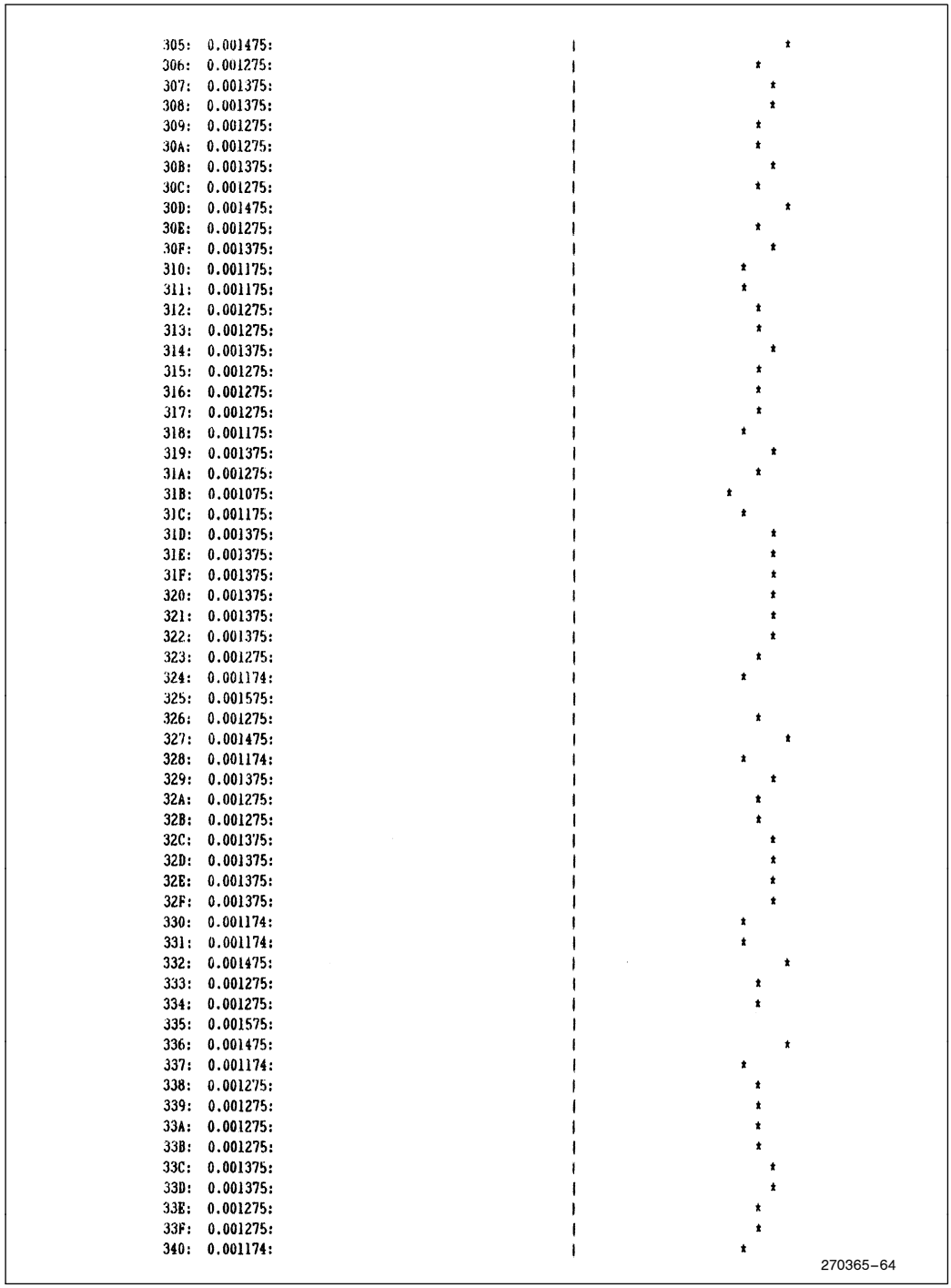


Repeatability Error, SN = 4130 (Continued)

```
2C9:  0.001375:                                                  *
2CA:  0.001175:                                              *
2CB:  0.001275:                                                *
2CC:  0.001275:                                                *
2CD:  0.001475:                                                   *
2CE:  0.001275:                                                *
2CF:  0.001175:                                              *
2D0:  0.001175:                                              *
2D1:  0.001175:                                              *
2D2:  0.001275:                                                *
2D3:  0.001375:                                                  *
2D4:  0.001175:                                              *
2D5:  0.001175:                                              *
2D6:  0.001275:                                                *
2D7:  0.001375:                                                  *
2D8:  0.001275:                                                *
2D9:  0.001275:                                                *
2DA:  0.001475:                                                   *
2DB:  0.001475:                                                   *
2DC:  0.001375:                                                  *
2DD:  0.001375:                                                  *
2DE:  0.001375:                                                  *
2DF:  0.001375:                                                  *
2E0:  0.001175:                                              *
2E1:  0.001375:                                                  *
2E2:  0.001275:                                                *
2E3:  0.001175:                                              *
2E4:  0.001175:                                              *
2E5:  0.001275:                                                *
2E6:  0.001275:                                                *
2E7:  0.001375:                                                  *
2E8:  0.001175:                                              *
2E9:  0.001275:                                                *
2EA:  0.001275:                                                *
2EB:  0.001175:                                              *
2EC:  0.001375:                                                  *
2ED:  0.001275:                                                *
2EE:  0.001275:                                                *
2EF:  0.001175:                                              *
2F0:  0.001275:                                                *
2F1:  0.001375:                                                  *
2F2:  0.001375:                                                  *
2F3:  0.001275:                                                *
2F4:  0.001275:                                                *
2F5:  0.001375:                                                  *
2F6:  0.001475:                                                   *
2F7:  0.001375:                                                  *
2F8:  0.001375:                                                  *
2F9:  0.001475:                                                   *
2FA:  0.001275:                                                *
2FB:  0.001175:                                              *
2FC:  0.001275:                                                *
2FD:  0.001275:                                                *
2FE:  0.001275:                                                *
2FF:  0.001275:                                                *
300:  0.001075:                                            *
301:  0.001275:                                                *
302:  0.001375:                                                  *
303:  0.001275:                                                *
304:  0.001175:                                              *
```

270365-63

Repeatability Error, SN = 4130 (Continued)

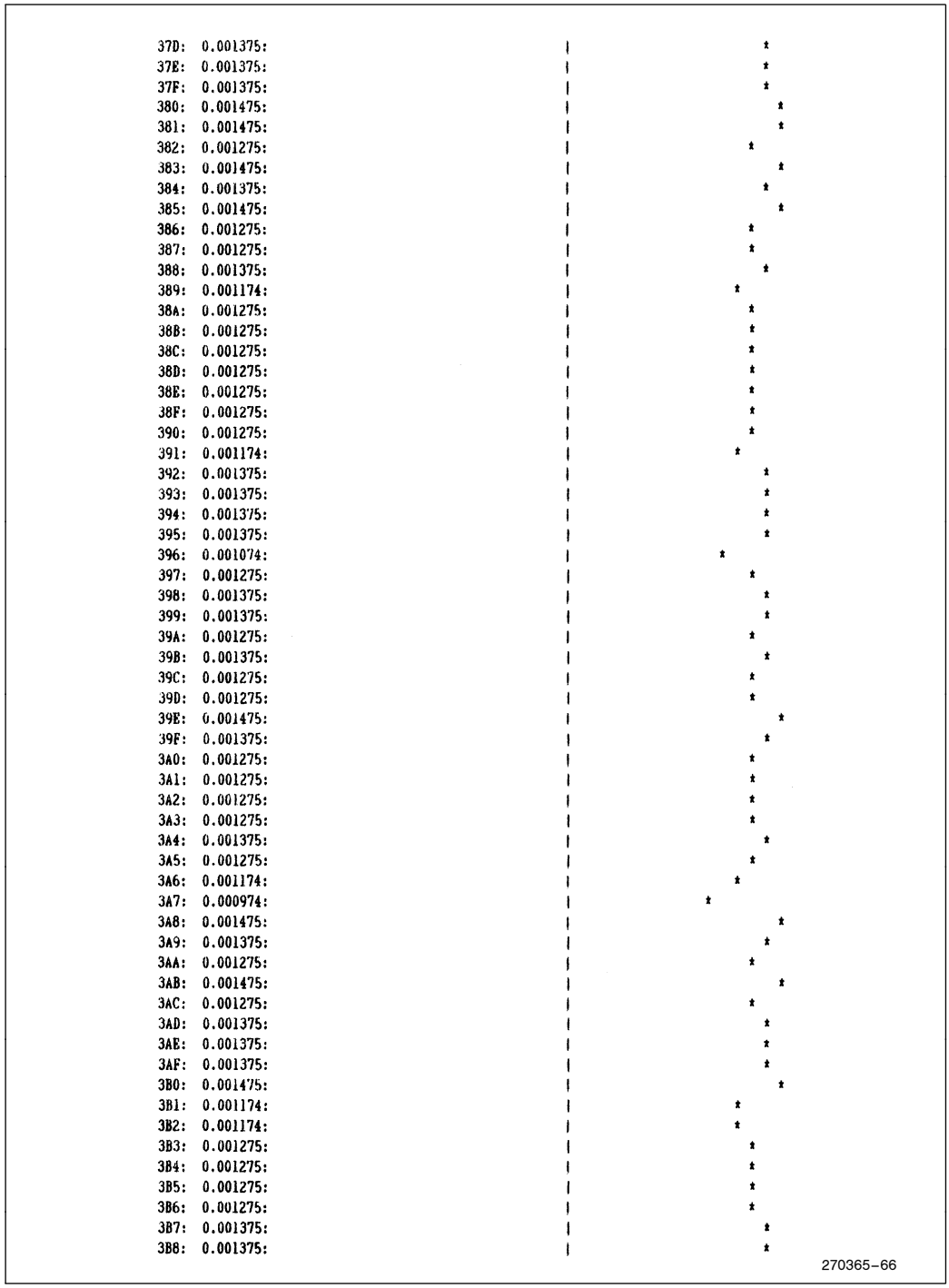


Repeatability Error, SN = 4130 (Continued)

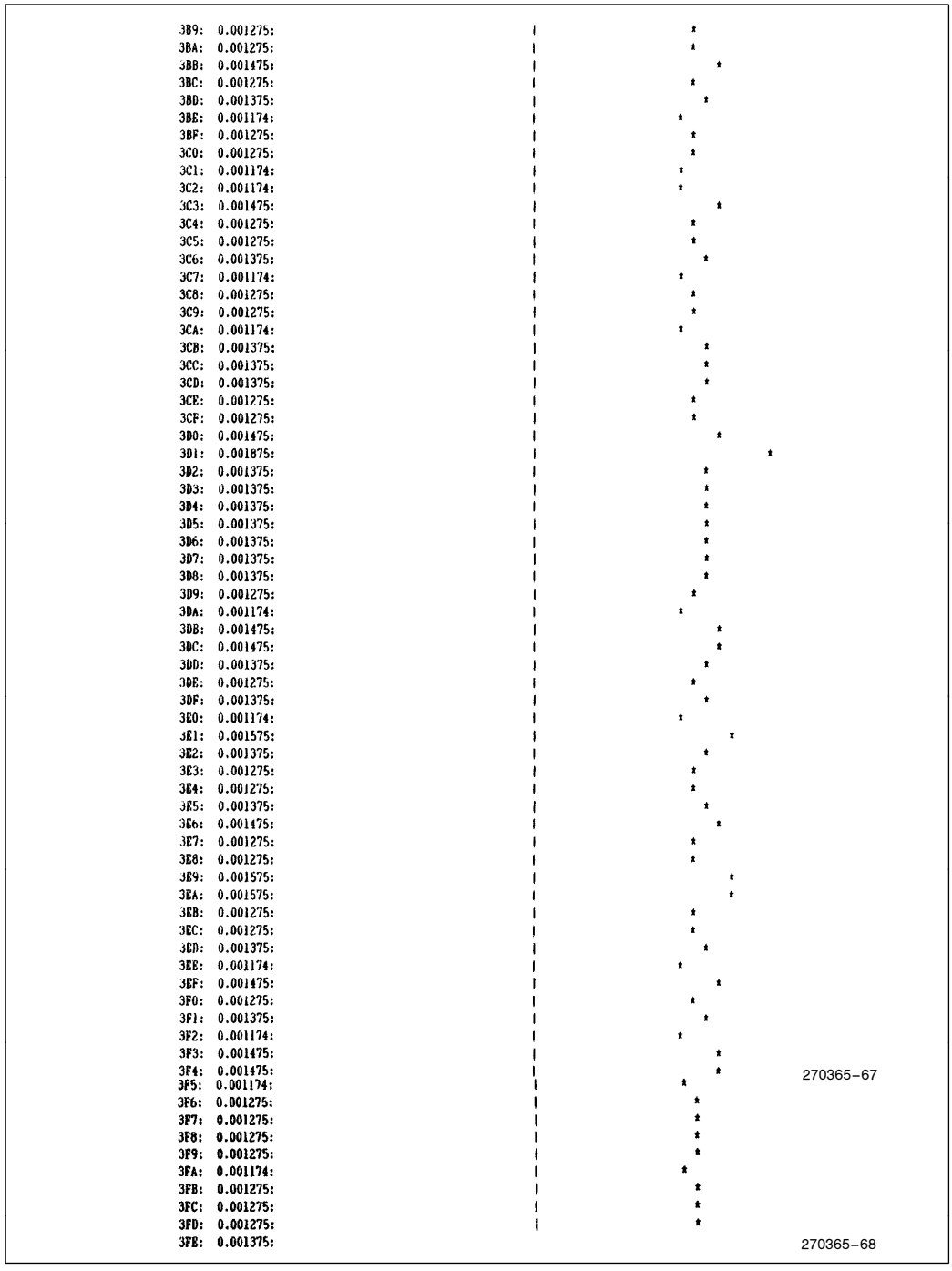
```
341:  0.001375:                                                    |                    *
342:  0.001375:                                                    |                    *
343:  0.001375:                                                    |                    *
344:  0.001174:                                                    |                 *
345:  0.001275:                                                    |                  *
346:  0.001174:                                                    |                 *
347:  0.001575:                                                    |                       *
348:  0.001375:                                                    |                    *
349:  0.001275:                                                    |                  *
34A:  0.001174:                                                    |                 *
34B:  0.001275:                                                    |                  *
34C:  0.001275:                                                    |                  *
34D:  0.001275:                                                    |                  *
34E:  0.001275:                                                    |                  *
34F:  0.001275:                                                    |                  *
350:  0.001375:                                                    |                    *
351:  0.001375:                                                    |                    *
352:  0.001375:                                                    |                    *
353:  0.001375:                                                    |                    *
354:  0.001275:                                                    |                  *
355:  0.001375:                                                    |                    *
356:  0.001375:                                                    |                    *
357:  0.001375:                                                    |                    *
358:  0.001074:                                                    |               *
359:  0.001275:                                                    |                  *
35A:  0.001074:                                                    |               *
35B:  0.001375:                                                    |                    *
35C:  0.001375:                                                    |                    *
35D:  0.001375:                                                    |                    *
35E:  0.001275:                                                    |                  *
35F:  0.001375:                                                    |                    *
360:  0.001174:                                                    |                 *
361:  0.001375:                                                    |                    *
362:  0.001275:                                                    |                  *
363:  0.001275:                                                    |                  *
364:  0.001275:                                                    |                  *
365:  0.001375:                                                    |                    *
366:  0.001375:                                                    |                    *
367:  0.001375:                                                    |                    *
368:  0.001275:                                                    |                  *
369:  0.001174:                                                    |                 *
36A:  0.001275:                                                    |                  *
36B:  0.001375:                                                    |                    *
36C:  0.001375:                                                    |                    *
36D:  0.001275:                                                    |                  *
36E:  0.001275:                                                    |                  *
36F:  0.001475:                                                    |                     *
370:  0.001375:                                                    |                    *
371:  0.001275:                                                    |                  *
372:  0.001375:                                                    |                    *
373:  0.001375:                                                    |                    *
374:  0.001275:                                                    |                  *
375:  0.001275:                                                    |                  *
376:  0.001275:                                                    |                  *
377:  0.001275:                                                    |                  *
378:  0.001275:                                                    |                  *
379:  0.001575:                                                    |                       *
37A:  0.001475:                                                    |                     *
37B:  0.001375:                                                    |                    *
37C:  0.001275:                                                    |                  *
```

270365-65

Repeatability Error, SN = 4130 (Continued)



Repeatability Error, SN = 4130 (Continued)



Repeatability Error, SN = 4130 (Continued)

APPENDIX E
BIBLIOGRAPHY

A/D Processing with Microcontrollers, Katausky, Horden, Smith

Apfel, R., et. al., "Signal-Processing Chips enrich telephone line- card Architecture". Electronics, May 5, 1982.

Analog Devices - Data-Acquisition Databook 1984, Volume 1

Blahut, Richard E., "Fast algorithms for digital signal processing", Addison Wesley Publishing Company, Inc., 1985.

Boyes, ed. - Syncro and Resolver Conversion, 1980

Brown, Robert Grover, "Introduction to random signal analysis and Kalman filtering". John Wiley \& Sons, Inc., 1983.

Burr-Brown Application Note, Testing of Analog-to-Digital Converters

Burton and Dexter - Microprocessor Systems Handbook, 1977

Candy, J., et. al., "The Structure of Quantization Noise from Sigma-Delta Modulation", IEEE Transaction on Comm. Vol. Com. 29, No. 9, Sept. 1981.

Candy, J., et. al., "Using Triangularly Weighted Interpolation to Get 13-Bit PCM from a Sigma-Delta Modulator", IEEE Transaction on Comm., Nov. 1976.

Electronic Analog-to-Digital Converters, Seitzer, Pretzl, Handy

Handbook of Electronic Calculations, Chapter 15, Analog-Digital Conversion

Harris Analog and Telecom Data Book

IEEE 162

IEEE STD. 746-1984

Intel Application Note AP-124 - High-Speed Digital Servos for Motor Control Using the 2920/21 Signal Processor

Intel Application Note AP-125 - Designing Microcontroller Systems for Electrically Noisy Environments

Irwensen, J., "Calculated Quantization Noise of Single - Integration Delta Modulation Coders" BSTJ Sept. 1969.

ITT Digital 2000 VLSI Digital TV System, MAA 2300 Audio A/D Converter, Edition 1983/9.

MIL-M-38510/135 June 4, 1984

MIL-M-38510/135 May 6, 1985

Modern Electronic Circuits Reference Manual

NBS Staff Reports, May/June 1981 P.22/23

Sheingold, ed. - Analog-Digital Conversion Handbook, 1972

Sheingold, ed. - Analog-Digital Conversion Notes, 1977

Sheingold, ed. - Non-Linear Circuits Handbook, 1974

Sheingold, ed. - Transducer Interfacing Handbook, 1980

Steele, R., "Delta Modulation Systems", Pentech Press Limited, 1975.

Taylor, Fred U., "Digital filter design handbook", Marcel Dekker, Inc., 1983.

Terminology Related to the Perf of S/H, A/D, D/A Circuits, IEEE Transactions