Registration Controller Firmware Release 1.6j dated 5 August 2002

Administration Update Manual

M L Riechers

Introduction

This document is an overview and description of the COMCO registration system, and the algorithms used to effect the registrations.

History

There have been something like eight major releases since the first version of this firmware was released August 1993. That first release incorporated Linear and Axial registration and Constant Error Correction, and only worked on Mark series presses with a 16.5 inch natural repeat.

Major milestones: In 1995 we added support to accommodate presses of different impression Cylinder sizes (i.e. 12.375 inch, 16.5 inch, and 27.5 inch Machine Basic Repeat Length, Commander and Mark series presses), Analox Roll sizes, and Encoder Counts per Impression Cylinder revolution. In 1999 we implemented the variable repeat option. In between, we implemented many small improvements and fixes.

This is the first attempt to document the operation of the registration control in a comprehensive way.

Scope

This document attempts to describe and explain the operation of the COMCO Automatic Registration System. This includes Linear registration, Axial Registration, the Constant Error Correction feature, and the Variable Repeat Correction feature, and the problems and considerations

that go with these things. We also touch on things affected by, or affecting the the COMCO Automatic Registration System, such as windowing, averaging, linear prediction, and operator choices. This document will not attempt to explain data entry, nor will it attempt to explain the operation of the host computer or the "motion controller" systems (e.g. automatic splice throw-off), except where pertinent to the operation of the COMCO Automatic Registration System,

Concept and Overview

The basic COMCO Registration System consists of an incremental shaft Encoder, a web photo sensor, a roll mark detector, a stepper motor which moves the print roll through a harmonic drive for longitudinal registration all attached to an electronic control. The electronic control directly supports an operator keyboard and display, and communicates with a central host computer over a RS-485 communications bus. In addition, there may be a stepper motor which moves the print roll from side to side, to support lateral, or axial, registration.

Each print station has its own, independent web photo sensor, roll mark detector, motor drivers, electronic control, and, for the Mark series of presses only, incremental shaft Encoder. (The Commander series of presses share a common incremental shaft Encoder.) Each station computes its own registration errors and corrections independently of the others, and is capable of being set up and run regardless of the operational state of any other print station, or the host computer.

In a nutshell, and without the complications of error checking, averaging, or windowing, the fundamental algorithm is this:

The incremental shaft Encoder measures the web travel, in units of roughly a thousandth of an inch, which the Registration Controller uses to keep a count, roughly analogous to an automobile's odometer reading,

of the web travel. When the Web photo eye trips, the Registration Controller records the web "mileage." Likewise, when the roll mark detector trips, the Registration Controller records the web "mileage." at that point, and subtracts the web "mileage." The Registration Controller compares this value with a previously established "set-point" value to develop a registration error. The Registration Controller then outputs a pulse-train to the stepper motor corresponding in number to the required registration correction.

To develop the Axial error, we use a (actually backwards Z) "Z" mark. We take the "mileage" at the top bar of the "Z," at the middle diagonal of the "Z," and at the bottom bar of the "Z." The "mileage" between the top and bottom bars is roughly constant, but the "mileage" between the top bar and middle diagonal, and between the middle diagonal and the bottom bar, varies in proportion to the side to side drift. So, the Registration Controller compares the two "mileages," and the result is the value of the Axial error. The Registration Controller outputs a pulse-train to the stepper motor corresponding in number to the required Axial registration correction.

To develop the CEC error, when the Web photo eye trips at the beginning of the next Repeat, the Registration Controller subtracts the previous web "mileage" from the current web "mileage," and compares that value with the value that the Repeat Length is supposed to be. The result is the value of the repeat elongation error — positive for too long, negative for too short. The Registration Controller computes the number of motor pulses necessary to correct to match the (ideal) print roll repeat to the web repeat, and spaces the motor pulses evenly over the following repeat.

The variable repeat feature is a variation of the Constant Error Correction feature. Whereas the Registration Controller computes the repeat elongation error from the measured length of a web repeat, the variable

repeat feature assumes an offset from a standard gear-pitch mandated repeat length, and uses both that offset and the repeat elongation error to compute the number of motor pulses necessary to match the specified web repeat. As in the pure CEC case, the Registration Controller spaces the computed number of motor pulses evenly over the following repeat.

System Assumptions and constraints

1. The unit of registration error is close to, but not, 1/1000 of an inch. This is dictated primarily by the commercial availability of incremental shaft Encoders at our desired resolutions.

For instance, the Mark IV press has a natural repeat of 16.5 inches. Incremental shaft Encoders are available at 5000 pulses per revolution, which, when used in quadrature, yields 20,000 pulses per revolution, but are not (or are at least in 1992 were not) commonly available at 4125 pulses per revolution. So the actual unit of registration error is .825 thousandths of an inch (16500/20000). Likewise, the actual unit of registration error is .859375 thousandths of an inch (27500/32000) for the Mark III, and the actual unit of registration error is .99 thousandths of an inch (12375/12500) for the Commander series presses.

2. We assume that the impression cylinder and the print roll are gear driven, and that the repeat circumference of the impression cylinder and print roll are roughly 2pi times their respective gear pitch radii. The reason is that all calculations are based on integer ratios of the natural repeat of the press to the print roll repeat. Currently both the numerator and denominator of this ratio is limited to 4095, which, for a Commander, means the gearing can't get any smaller than a 1/330 (.00302198) inch gear pitch.

There are advantages in this approach, both for (accurate) programming and for operator convenience. For instance, assuming 1/8 inch

gearing, the operator can set any of the valid 206 repeat lengths with at most seven and a half turns of the input wheel — and, since it's not possible to set an invalid repeat length, we don't have to check and reject invalid entries.

The fundamental gear pitch is the gear pitch of the impression cylinder. The program default gear pitch is 1/8 inch, but can be changed, and, a second, stand-by gear pitch can be programmed. (This stand-by gear pitch is most often found on dual-geared Commander presses, geared in both 1/8th inch and 32-diametral.) The impression cylinder gearing is simply the gear tooth count of the impression cylinder. For instance, these are the impression cylinder gear tooth counts for the current COMCO series presses:

Series	Repeat	Tooth Count
Commander	8 teeth/inch x 12.375	99
Mark IV	8 teeth/inch x 16.500	132
Mark III	8 teeth/inch x 27.500	220

While standard gearing for the Mark IV is a 132 tooth count, 32-diametral gearing for the Mark IV works out to a 168 tooth count, since fractional gear teeth are not allowed by physics. That means that within the allowable repeat length range of six to 36 inches, we have 305 possible print roll sizes, 6.08928571428571428571 to 35.94642857142857142857 inches in 0.09821428571428571428 inch steps. As a practical matter, the Registration Controller simply treats the whole affair as an integer ratio over the allowable range of 62 to 366 gear teeth ratio'd against 168 times the repeat length.

Our calculations accurately reflect the print that the press is capable of doing. However, operators are often confused when our calculations fail to match their diametral conversion charts in the fourth, fifth, or sixth decimal place: it is hard to convince them that most likely they don't

match because of round-off error or assumption on the part of the chart maker, the difference is inconsequential, and in any case what they'll get is what the press prints, not what the chart says the diametral converts to. Oh, well.

3. For the moment, we have made a tacit assumption that all machines on which we'll be running have an integer repeat relationship of some multiple of 1.375 inches:

Series	Multiple	Repeat
Commander	9 x 1.375	12.375
Mark IV	12 x 1.375	16.500
Mark III	20 x 1.375	27.500

However, this is a bit weak, and will probably not long survive.

4. At the moment, we assume that the movement of four motor pulses equal the web travel of one encoder pulse. For instance, if we have a measured error of five encoder pulses, then we will correct by issuing 20 motor pulses in the opposite direction, assuming unity gain. However, we can easily change this to accommodate any reasonable ratio.

Doing and counting Motor Pulses

After the Registration Controller calculates a Registration Error, it "outputs a pulse-train to the stepper motor corresponding in number to the required registration correction," and, the Registration Controller computes the number of motor pulses necessary to correct to match the (ideal) print roll repeat to the web repeat, "and spaces the motor pulses evenly over the following repeat." Those assertions belie the complexity involved in the correctional movement, however.

First, the stepper motor, gear train, and print roll have mass, and therefore present an inertial mass to accelerate or decelerate; although we'd like to, we cannot do the Registration Corrections instantaneously. So, the Registration Controller keeps a count of the number of stepper motor steps that "should have been done already," times the interval between issuing motor pulses according to whether it needs to accelerate or decelerate the motor, and, decrements the step count each time it issues a motor pulse.

Until press speeds get really, really fast, — much faster than they are today — the acceleration requirement is not an issue for straight registration correction. However, problems do arise when "blended in" with the CEC corrections when the CEC rate of correction is very high. It is very easy to overcorrect for Registration Error when the Registration Error is opposite in sign to the CEC elongation, but difficult to "catch up" to the CEC subsequently.

Second, the CEC requires that its corrections be spaced evenly around the repeat. At first blush, this seems simple and straight forward: just take the repeat length in encoder counts and divide by the number of motor steps needed to correct to the measured CEC, and, every time the press moves that number of encoder counts, issue a motor pulse. This generally works fine when we have both a slow press speed and small CEC corrections. However, there are two problems: 1), CEC motor steps must be integrated with the Error Correction motor steps, for they can interfere with each other, particularly at the beginning of the repeat (or, more properly, just after the Registration Error has been read), and 2), at higher press speeds or greater CEC corrections the motor must be properly accelerated, or the motor will stall, ruining the Pressman's whole day.

So, we use two CEC strategies. At low press speeds and low CEC corrections, we use the pulse spacing method. Otherwise, for high press

speeds or high CEC corrections, we iteratively calculate a step rate to match the CEC correction to the current press speed, and then attempt to run the stepper motor at that speed. In either case, we use the motor steps that "should have been done already" bucket to account for, and do, the Registration Correction motor steps. If we're speed matching, the motor steps that "should have been done already" value serves to adjust the target motor speed higher or lower than nominal to attempt to zero this value.

Third, the Registration Correction motor spins the Print Roll. Really, you say, in an ironic sort of way. Yes, I say, and that presents a measurement problem.

Imagine that the Registration Controller takes a Print Roll Mark encoder reading and that reading is (for the heck of it) 2345, the Registration Set point is 10,000, and the Repeat Length is 26667 encoder counts (22.000*in.*). That means that we expect to see the Roll Mark at encoder count 12345.

So, now monkey-wrench the thing: suppose that while the press journeys from encoder count 2345 to encoder count 12345, the Registration Controller throws in a correction of some kind — it matters not what the correction is for, but let's say for CEC — say to the tune of 100 motor pulses in the plus direction. This has the effect of moving the Roll Mark 25 encoder counts (100/4) closer to the Web Mark — so we ought to either subtract 25 encoder counts from the Roll Mark Encoder Counter, or add 25 encoder counts to the Web Mark Encoder Count that we should read at or about encoder count 12345, before calculating the error.

In practice, the Registration Controller does the latter. In theory, we should scale the motor count due to CEC or Variable Repeat to the elongation factor, and leave the motor count due to Registration Correction unfactored, but in practice this scale is almost always well under 1%,

and seems to make no practical difference. In our example the correction would be probably about .0625 encoder count, with which we cannot deal, because it's way too small.

How and when we discard Registration Error Readings

First, we acquire a viable encoder count at the Web Eye Mark photo sensor event. (At this point, if we're doing windowing, then we check if we've also got a Roll Eye Mark photo sensor event within the last repeat plus 512 encoder counts, and if we don't, then we discard this reading.)

Next, we compute the registration error. (If we're not doing windowing, and we haven't got a Roll Eye Mark photo sensor event within the last repeat plus 512 encoder counts, then we set a flag saying the error's no good, and for the purpose of calculating a CEC, set the error to zero.) The formula is:

$$(WbEyCnt - RlEyCnt) - roundup(cSkewA - MatRlEy)/4$$

which is the encoder count at the Web Eye minus the encoder count at the Roll Eye minus one fourth of the number of motor pulses commanded between the Roll Eye and Web Eye rounded up. Now, the result could be less than zero, so, if it is, we add a repeat length.

We take this value, and from it subtract the Registration Set point. The result can be positive or negative, but we want to ensure that the error lies within one-half a repeat length plus or minus. So, for instance if the error is greater than positive one half repeat length, we subtract a repeat length to get the proper negative error.

Next, if windowing is in effect, we ensure that the error lies within

the windowing zone.

Now we see if the error is outside the Out of Tolerance Noise Band, where OOTNBAND is:

According to the formula: 5 times the DEADZONE times 5/8 of the Error Array, then doubled to make in terms of 1/2000 inch pulses.

More concretely,

if error averaging, and $DEADZONE > 0$, The Out of Tolerance Noise Band is:

$$10 * DEADZONE * erroraveragingsamplesize * 5/8$$

if error averaging, and DEADZONE = 0, The Out of Tolerance Noise Band is:

10 * error averaging sample size * 5/8

if not error averaging, and $DEADZONE > 0$, The Out of Tolerance Noise Band is:

10 * DEADZONE

if not error averaging, and DEADZONE = 0, The Out of Tolerance Noise Band is:

10

Now our actions depend on whether or not we've accepted a Web Eye Mark and computed a registration error in the last Repeat. If we have not, then we look at our just computed Registration Error. If the error is outside the Out of Tolerance Noise Band, then we totally abandon this

web mark and all info concerning it. If it is inside, we accept the error, but skip calculating a CEC, note the encoder count at this point. We're probably protecting a web shift, WbEyShft to current WbEyCnt, and clear WbEyMsd.

If we have accepted a Web Eye Mark and computed a registration error in the last Repeat , we calculate a mark to mark distance to see if we should accept the reading, but if our computed error is outside the Out of Tolerance Noise Band, we set a flag to indicate that the Web Eye Mark pattern "shifted" from its regular Repeat. Otherwise, the error is within the Out of Tolerance Noise Band. In either case, calculate is as follows:

We subtract the present web eye encoder count from the previous web eye encoder count, and from that subtract the repeat length. If the result falls between -127 and 128, accept the reading, and go on to calculate the CEC. (There used to be a further test to see if the repeat length error lay between 5/1024 of a repeat, e.g. a 16 inch repeat would restrict the test to plus and minus 78 on Commanders and plus and minus 94 on Mark Fours. That test went by the boards when it interfered with the infinite repeat feature.)

Otherwise, the repeat length is much to small or large, and it is possible that this is either a spurious reading, or that the web mark has shifted. If the mark is spurious, then presumably the correct mark should show up in approximately the correct position with respect to the last good mark read. If the mark has shifted with respect to the repeat, and if this is the first shifted mark that the paper eye has seen, then repeating the test above fails. Otherwise if the mark has shifted, and if this mark falls about one repeat length from a previously recorded mark, then we want to accept the new mark.

Now, there might be other spurious marks in the track. If the web

really has shifted, we want to identify the correct one. So, we have a flag set if we are "Protecting a web shift," which if on at this point, we consider this mark noise, and cast it out. Otherwise, we subtract the present web eye encoder count from the last time we accepted an eye encoder count, and from that subtract the repeat length. If the result falls between -127 and 128, we accept the reading, and go on to calculate the CEC. (There used to be a further test to see if the repeat length error lay between 5/1024 of a repeat, e.g. a 16 inch repeat would restrict the test to plus and minus 78 on Commanders and plus and minus 94 on Mark Fours. That test went by the boards when it interfered with the infinite repeat feature.) If the last eye encoder count was "good", then this test will fail, because it merely repeats the test above.

If the test fails, then we record this present web eye encoder count as the last time we accepted an eye encoder count, set the we are "Protecting a web shift" flag, and do nothing else with this mark. (XXX WebShftP is cleared *only* by one repeat going by from the last WbEyShft point. WbEyShft is *only* set in this error routine and if the upc takes a good eye mark IFF WebShftP is clear). If a good mark is found in approximately the place we expect to find it, then we accept that mark. Otherwise, we wait for one repeat length minus 1811 encoder counts before clearing the "Protecting a web shift" flag and, if another mark is found, repeating this test.

After we calculate the CEC or not, as above, if we're doing error averaging, and if the error is OK (remember above, we may have set a flag saying the error's no good,) we enter this error into the traveling averaging array, and calculate the average.

Next, if the error is OK, we calculate the eight bit error, and limit the error to less than than or equal to 127 but greater than than or equal to -128. If the error is less than the DEAD ZONE, then treat the error as if it were zero. But if the error is greater than 63 or less than -64, and

the Small Corrections option is not in effect, process the error as a "Big Error:" "Dump" the error averaging array, include any CEC in the error itself, kill any CEC going on, and use the raw (Big) error to correct.

If the error is less than or equal to 63 but greater than or equal to -64, (or if Small Corrections are in effect), process the error as a normal error by scaling the error by the gain, and, if we're in auto registration mode, and regardless of whether or not we may have set a flag saying the error's no good, set up this Repeat's CEC to the motors, if it's appropriate.

Now if we still have a correction, and if we're in auto registration mode, we start the motors to correct the error, and, if we're doing averaging, revise each element in the averaging table by the correction we're about to make, under the theory that by physically shifting the Roll Eye to Web Eye relationship we are also shifting every error reading we've made in the past.

We're done.

CEC elongation calculation

The default behavior of the CEC calculation part of Registration System is that it is always enabled. In fact, CEC calculation cannot be turned off. As long as the press is running, the Registration Controller calculates a CEC elongation.

The CEC elongation figure is calculated as a variance to the nominal repeat length, in encoder counts. For example, for a 22.000*in.* nominal repeat, a +20 CEC elongation would indicate that the repeat is long by 20 encoder counts (i.e. the measured repeat is 22.000*in.* + 20 encoder counts), and a -20 CEC elongation would indicate that the repeat is short by 20 encoder counts (i.e. the measured repeat is 22.000*in.* - 20 encoder counts). On a Mark Four press, since an encoder count equals

.825 thousandth$in.$, the actual throw lengths for the + and - 20 encoder counts will be 22.0165$in.$ and 21.9835$in.$.

To figure the CEC elongation, by default the Registration Controller keeps a list of the last 29 measured throw length errors, and all 29 participate in the linear regression. The operator may select fewer than 29, however.

The algorithm works this way: just after the registration error has been taken, and the system read a good Web Mark previous to the just read Web Mark, the Registration Controller figures the Throw Length error, and if it's reasonable, places it into the Linear Regression Array. Then it figures a best fit line using the Method of Least Squares of the last n repeats

$$y = ax + b$$

where y is the Throw Length error and x is the repeat, 1 being the earliest, and n being the current repeat, according to the slope

$$a = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$

and the y intercept

$$b = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

and predicts the $n+1$ Throw Length error (y) according to:

$$y = ax_{n+1} + b$$

Of course, there is a simplifying predictive formula, which takes advantage of the fact that all x are sequential. Otherwise, the math would overwhelm the microprocessor.

Why use a linear regression instead of a simple average of the last n repeats? Because we are interested in predicting the throw length of the

next current repeat, for that is the one we'll be applying the correction to. If the throw length is unchanging, then it doesn't matter whether we use averaging or the linear curve fitting. However, if the throw length is gradually increasing or decreasing, then the average throw length will be in error, while prediction using linear curve fitting should give accurate results.

The accuracy of the predictive linear curve fitting depends upon the quality of the print: the precision of the underlying machinery (how well the press is made) on which the job is being run, the quality and consistency of the web material, and how well the press is set up (you gotta get the tensions right). And, of course, if this is a re-insertion print job, the precision of the previous press and how well it was set up. Bad print might manifest itself as regular, periodic fluctuations in the throw length, which might be graphed as a sine wave, or a series of sine waves, which may or may not be predictable from the information that the Registration Controller has to work with. Or, bad print may show up as essentially random variations in throw length. Be that as it may, the effect of bad print on CEC curve fitting is disastrous: the prediction could be high, low, or anywhere in between. A simple throw-length average would be better, but, then again, bad print is bad print.

Usually, in practice, the fly in the ointment for CEC is bad web tensions. But in fact, we almost always see good results from the CEC, no doubt resulting from the fact that the same job set-ups that produce good print ("the art of printing") also produce good CEC. So, it works well enough, right?

Well, there exist measures of "goodness of fit," which would indicate how much confidence to place in our CEC prediction. To the operator, high confidence would indicate that his print is going well. Low confidence should mean that Operator Activity is in order: Do Something To Make It All Better, or scrap the job.

However, calculating the "goodness of fit" is a good deal more intense than calculating the CEC linear prediction, and figuring out an effective and meaningful way to present the "goodness of fit" to the operator, so that he can use it to advantage, is a bit of a challenge. Nonetheless, we recommend implementing the "goodness of fit" as a needed and valuable future enhancement to the system.

CEC elongation calculation and "Repeats Around"

We've had in mind a certain feature for the Registration Systems since we fielded the first one, but have not yet implemented. It is called Programming Repeat Ratios, and, for completeness's sake, I'd like to describe it here.

Occasionally, Printers want to match Repeat Lengths bearing integral repeat ratios to each other. For instance, imagine that we're printing "stamps" on the web, one inch apart, linearly, and equally spaced. Station One is printing with a nine inch repeat Print Roll, while Station Two is over-printing with an eight inch repeat Print Roll. Station One has nine images around, while Station Two has only eight. If the tooling is precise, however, the images will match to the extent of the "stamps" hitting one on top of another, although the "eight" will progressively "walk around" the nine-repeat image for a cycle of nine repeats. I'm told Printers often do this to vary the composition of the images — in this case, the 72 stamps varied over the nine repeats.

Now note that the two stations bear a definite integer relation to each other, and that number, unreduced, is 72. In every situation that I'm aware of, where Printers would like to do the like of the above, or where they are forced into such a situation, there exists such a "common denominator." The number may be rather large, but it always seems to exist.

If the tooling is precise, indeed, and the repeat pattern is carefully controlled to evenly space the images, then this does not generally present a problem to the Registration System. However, there are certain outrećonditions which cry out for help.

Imagine the following scenario. A Printer prints his "stamps" on an offset press. The "stamps" are spaced .870*in.* apart, linearly, and equally spaced through a 30.450*in.* repeat: 35 "stamps." (At least the Printer *claims* a 30.450*in.*, and *not* a 30.500*in.* repeat). He sets the run aside for a couple of weeks and allows awful things to happen to the run — no, back up, correction, correction, that isn't really necessary, although it makes the story more juicy. Well, so he puts it aside, then runs the web on a COMCO press — er, die station, with Registration Control and CEC. Only it's a rotary die, with a 21.750*in.* repeat: 25 "stamps" around.

Now to get around the obvious difficulty of trying to reconcile 30.450*in.* registration marks with a 21.750*in.* repeat, as the die station sees it, registration marks are printed co-incident with the "stamps." That is, when it printed the "stamps," the offset press also printed 35 registration marks spaced .870*in.* apart linearly for the benefit of the COMCO re-register. The COMCO Press Operator then sets Registration Mark "windowing" in effect to select one of the 25 Marks to register on.

Problem: the selected Mark is not the "same" Mark from repeat to repeat. In fact, from the offset press's point of view, the Mark will always be 10 Marks (8.700*in.*) short of a full repeat — and, to calculate the throw length error, it is the *offset press's repeat that counts.* So sometimes the Registration Controller is measuring the (shall we say wrong?) throw length within the offset press's repeat, and sometimes it's measuring between repeats, but always between different Marks. On good days when the gods smile and the offset printing is perfect this is OK, but occasionally this is bound to come to a bad end, and it does. When there are Mark to Mark variations, or there is a little space left between

(the offset press's) repeats, the Registration Controller sees essentially random measurements, which drives it bananas.

Solution: Have the offset press lay down just one registration Mark per each of its repeats, and have the Registration Controller "travel" the Registration Set Point, and the expected "Next Mark," forward on a 7/5ths of a repeat ratio. The programming to do the actual "traveling" is relatively trivial; the biggest hurdle is defining, documenting, and communicating how to do these things to our customers, (which we have dubbed "Repeats Around"), and writing the operator entry programming. (There has always existed a key, albeit unimplemented, on the Operator's station Keypad for "Repeats Around"). I have sketched a scheme to support Programming Repeat Ratios; it is as straight forward as it can be, but might be a bit tricky for some operators to cope with. However, I don't think that customers who dream up deliberately overprinting on 9-8 ratios (and greater) will have any difficulty mastering it.

Well, I gave a couple of examples of matching dissimilar, but related by integer ratio, repeat lengths. I don't think that this type of thing is going to go away; on the contrary, I think that it will continue to offer us "opportunities" in the future.

Saved Variables

The Registration Controller saves and uses the following variables:

1. Model number of this particular machine.
2. Number of teeth on the Impression Cyl,Gear-side,
3. Number of teeth on the Impression Cyl,Operator-side.
4. Station address (used as 0-31, but operator sees 1-32),

5. Current station type, 0=motion, 1=registration.
6. Operating modes 1,
7. Operating modes 2,
8. Operating modes 3,
9. Operating modes 4.
10. Number of teeth on the Print Roll.
11. Distance from eyemark for EYE Mark back window,
12. and Front window set-up.
13. Negative distance from eyemark for Front window
14. IZONE inspection.
15. Positive dead zone
16. Compensator gain, converted from host 0-9 to 11-128.
17. Sample Size for Error Averaging.
18. Sample Size for Error Averaging, Axial.
19. Error Zone Set-Point.
20. The Calibrated operator side Pot Limit, in A/D units.
21. Linear Registration Set-Point,
22. Axial Registration Offset Set-Point.
23. where to store the PRDrSel bit.

24. Sample Size for Linear Regression, CEC.

25. NV Web Distance from the First station: 1st component,

26. NV Web Distance from the First station: 2nd component,

Normal operation

There are two conditions of operation: the press is stopped, or, the press is running. The Registration Controller determines which by timing the Encoder pulses; normally, the Registration Controller decides that the press is running when the press speed exceeds something over 35 feet per minute, and decides that the press is stopped when the press speed falls below something under 35 feet per minute. However, the operator can choose 5 feet per minute as the discriminator.

When the press stops, the Registration System normally finishes any registration corrections it was making, and becomes inactive. While the press is decelerating from 35fpm to 0fpm the Registration Controller normally continues to issue CEC and Variable Repeat motor pulses, continues to read the encoder, continues to read the web and roll marks, but will not respond to them nor will it calculate any new registration errors. The Registration Controller clears the averaging history array, and the linear regression history array.

The operator is permitted to enter most operating values and parameters, such as Repeat Length, at any time; however, usually the best time is when the press is stopped. Save-able parameters are immediately saved to non-volatile memory. Also, the press must be stopped to execute Pre-Registration.

When the transitions from stopped to running, the Registration Controller starts or resumes calculating registration errors and CEC elongations, and, if the Variable Repeat option is in effect, immediately resumes

issuing motor movements to adjust the print roll to where it should be in the repeat. If averaging is in effect, it starts re-filling the averaging array(s), and the linear regression array. If the system is in Automatic Register, the Registration Controller makes registration corrections as soon as it reads its first registration error. However, if CEC is in effect, the Registration Controller waits until at least four throw values have been read before issuing CEC corrections. This is to give the linear regression algorithm a chance to stabilize.

Regardless of whether the press is stopped or running, the operator may place each of the linear registration, axial registration, or CEC subsystems in "manual" or "automatic." (Obviously, if the press is stopped, then it makes no operational difference whether the subsystem is in "manual" or "automatic," and the only consideration is to have the proper mode when the press starts.) In "manual" means that the subsystem will calculate errors, but will not make corrections. In "automatic" means that the subsystem will calculate errors, and will also make corrections based on those calculations.

While the press is running and adjusted for proper registration, it is usual for the Operator to select "automatic linear registration," and to select "linear registration" from the keypad. This makes it easy to make linear registration adjustments. The operator entry wheel is "pointing" to linear registration adjustments, so rotating the wheel "up" a click will adjust the registration set-point "up" a thousandth of an inch or so, and simultaneously move the print roll "up" the same amount of distance with reference to the web print image: the image that this station is printing will shift "up" in relation to the print images laid down by the previous print stations, and the Automatic Register Correction will keep it there. Similarly, if the operator twirls the adjustment wheel "down" a whole rotation, about 32 clicks, then the print image will shift "down" about 30 thousandths, and the Automatic Register Correction will keep it there.

If the Operator to selects "manual linear registration" and "linear registration" from the keypad, then his adjustment wheel will shift the print roll image "up" or "down" in relation to the print images laid down by the previous print stations, but this will not change the registration set-point, and, since it's turned off, Automatic Register Correction will do nothing, obviously. This feature is most often used to bring the image into rough initial registration when proper web tensions have yet to be established, or when the operator wishes to deliberately, briefly, bring the system out of registration and back in.

The operator uses the axial registration in much the same way as linear, except that, whereas there are no limits on how far he can adjust the linear register, (if he rotates the wheel 1000 times in linear register it merely adjusts into the following or preceding repeat(s)), there are side to side limits. The physical travel limit is about 20 millimeters from side to side, and is dictated by the physical construction of the press. We further limit the travel to about 16 millimeters. It is important that the Operator physically sets up the web and guides initially so that the possibility of his electrical adjustment starts roughly in the middle of the 16 millimeter adjustment. If he does not, then he will likely run out of adjustment room.

If the Operator selects "automatic axial registration," and selects "axial registration" from the keypad. then the operator entry wheel is "pointing" to axial registration adjustments, so rotating the wheel "right" a click will adjust the registration set-point "right" a thousandth of an inch or so, and simultaneously move the print roll "right" the same amount of distance with reference to the web print image: the image that this station is printing will shift "right" in relation to the print images laid down by the previous print stations, and the Automatic Register Correction will keep it there. Similarly, if the Operator to selects "manual axial registration" and "axial registration" from the keypad, then his

adjustment wheel will shift the print roll image "right" or "left" in relation to the print images laid down by the previous print stations, but this will not change the registration set-point, and, since it's turned off, Automatic Register Correction will do nothing, obviously.

By default, CEC is enabled whenever Automatic Linear Register is selected, but the Operator can turn it off. And, like the linear and axial subsystems, CEC always calculates a throw-length error, regardless of whether or not it is in Automatic mode. CEC cannot be turned on (i.e. produce error corrections) unless Automatic Linear Register is also selected, however. Other than tangential adjustments to the size of the CEC linear regression array, there are no adjustments that the operator can make to the CEC — it is either on or off. The Operator can choose to display either the CEC or the Axial Error on the second line of the display, however.

Saving values in non-volatile memory

The Registration Controller has a limited amount of non-volatile memory, technically known as Electrically Erasable Programmable Read-Only Memory (EEPROM). So the Operator won't have to keep re-entering data each time power is removed from the system, the Registration Controller saves all parameters and critical data in the EEPROM.

Unfortunately, the EEPROM is write-life-limited. We can reasonably expect 100,000+ write cycles on any byte in EEPROM before it "wears out," but the manufacturer only guarantees any byte in the EEPROM for 10,000 write cycles. This is because the manufacturer must guarantee in worst-case conditions, which include operation above 200 degrees Fahrenheit, which is really hard on EEPROM. Be that as it may, we try to design to the 10,000 cycle limit.

New data and parameters that the Operator enters, or are received

from the host computer, while the press is stopped, are written immediately to EEPROM. The theory here is that this data is in earnest: since the press is stopped, the Operator isn't changing things just to see what will happen; obviously, nothing will "happen" until the press is running.

However, the Operator may very well change things while the press is running "just to see what will happen," and that is quite legitimate. He may, for instance, change the Dead Zone, or the Averaging, or the Gain, to see what effect that has on print quality. If we re-wrote the EEPROM every time, it would quickly wear out. So, the Registration Controller waits to write the new values until the press stops.

The registration set-point values are a special case. The operator changes the linear set-point value, for instance, any and every time he moves the Entry Wheel, as long as Linear Auto is selected, regardless of whether the press is stopped or running. The Registration Controller simply waits about ten minutes to write the new set-point value. Thus, and because we hope on average that things will "settle down" and have a smooth run, a new set-point value is likely to be written only two or three times per print run.

There is some confusion on these points. Our customers will occasionally test whether values are being saved, and return the controllers to us as defective, because they forgot or do not realize that they must "halt" the press or simulator before they power-down the controller.

Modes of operation: value selected

The Registration System is capable of running in a number of different modes of operation. Some of these modes are entered by selecting a value, while others are selected by specifically turning on a feature. Major modes selected by selecting a value are:

1. Dead Zoning. The default behavior of the Registration System is to make a registration correction whenever it computes a non-zero Registration Error, and "Automatic Register" (see above discussion) is in effect. The operator can change this by setting a non-zero "Dead Zone," which "turns off" making corrections to Registration Errors which absolute values are less than or equal to the Dead Zone value. This is useful to suppress small corrections in response to insignificant errors, which might themselves actually induce a random, but real, registration error.

The Dead Zone value is compared to the Registration Error after averaging, but before any Gain is applied.

1. Gain adjustment. The default behavior of the Registration System is to make exactly enough Registration Correction to correct the Registration Error. We never want to make more Correction (over-correct) than warranted, because that invariably sets up positive feedbacks and wild registration oscillations. But, we often want to under-correct, to damp any incipient registration oscillations.

The operator can dial in a Gain value from 1 to 10, where 1 means that the Registration Controller will apply only 10% of the Registration Correction to correct the Registration Error, 5 means that the Registration Controller will apply only 50% of the Registration Correction to correct the Registration Error, 7 means that the Registration Controller will apply only 70% of the Registration Correction to correct the Registration Error, and 10 means that the Registration Controller will apply 100% of the Registration Correction to correct the Registration Error, with intermediate values likewise.

However, applying a Gain factor to Large Errors is unjustified, so We Don't Do That. If the value of the absolute error is greater than or equal to 64, it is a Large Error, and Large Errors ought to be corrected as soon

as possible. So, the Registration Controller "dumps" the moving average array (explained below), and, if "Automatic Register" (see above discussion) is in effect, immediately uses the full value of the Large Error to make the correction, with full Gain. The notion is to get the Registration Error within the +-63 thousandth window as quickly as possible.

Assuming the error is Small, Gain operates on either the most recent Registration Error read, or the Averaged Error if Registration Error Averaging is in effect. Since it's possible to compute fractional errors from the Gain, we take advantage of the fact that it takes four motor pulses to move one unit of error by computing the Gain'ed error four times larger than it should be, rounding the result toward zero, and using that as the motor correction. That means that the Registration Controller actually corrects to one quarter, half, three-quarter, or a whole (putative) thousandth of an inch. This probably doesn't imply significantly greater accuracy, but the smaller movements permitted does suggest a "smoother" overall registration — i.e. less chance for over-correction.

After all this, the Registration Controller arbitrarily subtracts two motor pulses from the proposed motor movement, and, if a pulse count remains, starts the correction motor. Otherwise, the motor is shut down, at least with regard to Error Correction.

The rationale for arbitrarily subtracting two motor pulses from the proposed motor movement is this: if we measure, say, an error of 6, then we assume that the actual error is not under 5.5, nor is it over 6.5. But, it could be 5.5, in which case any correction greater than 4 times 5.5, or 22 motor pulses, is an over-correction. So, we subtract the two motor pulses to ensure that we always make the correction .5 less than the putative error, thus ensuring that, in theory, we usually slightly under-correct, but never make an over-correction.

1. Registration Error Averaging. The default behavior of the Registra-

tion System is to use and display just the last read registration error. However, both the Linear and Axial registration can be configured to "average" the last read registration error with an operator selectable number of previous registration errors. To this end, the Registration Controller keeps an array of the last ten registration error readings. Therefore, the operator may select up to ten readings to be included in the moving average, — the current reading, and the nine previous.

The algorithm works this way: just after the registration error has been taken, and the absolute value of the registration error is less than 8192, the Registration Controller inserts the error at the end of the list of ten last read registration errors, and increments a value representing the number of "participating" registration errors to be averaged. If the resulting value is not greater than the operator selected number of readings to average, the controller uses that value as the number of readings to average; otherwise, it uses the operator selected number. The reasons for this will be apparent in a minute, but for now, call this the last n readings.

The Registration Controller adds together the (signed) last n Registration Error readings, and divides the sum by n, which yields the registration average. That average is set aside to use for the subsequent registration error correction and to display to the operator and to send to the host computer.

Since the Registration Controller actually corrects to one quarter, half, three-quarter, or a whole (putative) thousandth of an inch, to facilitate back-correcting the averaging array, explained below, the values in the error averaging array are kept as values four times larger than the actual raw errors. This doesn't imply significantly greater accuracy, but the smaller movements permitted does suggest a "smoother" overall registration — i.e. less chance for over-correction due to round-off. Be that as it may, let's be clear that the averaged error that the operator

sees and is sent to the host is divided by four and rounded to the nearest putative thousandth.

The purpose of Registration Error Averaging is to "average out" random errors such as early or late photo eye triggers, and, to some extent, regular errors such as those induced by excentric gearing, spring, and torsional effects. As such, averaging Large Errors is unjustified, so, beyond calculating one Large Error, We Don't Do That. If the absolute value of the averaged error is greater than or equal to 64, it is a Large Error, so the Registration Controller "dumps" the moving average array (explained below), so that the next reading will *not* be averaged, and, if "Automatic Register" (see above discussion) is in effect, makes the correction with full Gain. If the station is in "manual," and the Registration Controller is reading mostly or all Large Errors, then an interesting side effect happens: the Error Averaging will actually not, in fact, happen on subsequent repeats, even though the station is "in" this mode.

Now if the station is in "manual," nothing further happens with regard to registration (i.e. you can stop reading this section here, but skip to the discussion on "dumping" the error averaging array).

If "Automatic Register" (see above discussion) is in effect, the Registration Controller immediately begins the corrective motor movements to the Print Roll, which will be continued and sustained automatically. Now, the Registration Controller must apply a correction to the last n raw readings in the moving average array, and this is a bit tricky to explain.

Unless an upset happens, the Print Roll will have actually moved to the corrected position by the time the next registration error is taken. So, with regard to the *next* registration error reading, the error that we just added to the end of the moving average array — the "last read registration error," — is "wrong" by exactly the amount of correction that was done.

To illustrate the point, imagine that we're doing a moving average of two error readings. That means that we'll read the current registration error, and average it with the last one taken. Suppose, then, that the Registration Controller takes an error reading of -40 thousandths, does its averaging thing, and determines that it must make a (a large, for the purposes of argument) +45 thousandths of an inch registration correction. Lo and behold, the +45 thousandths correction is dead-on perfect, and the *next* registration (raw, unaveraged) reading is zero. What do we average?

Intuition would suggest that if we average the previous -40 error and the current 0 error, the resulting +20 correction would be way too much — an over-control. The previous -40 error should be corrected by the +45 thousandths motor movement. If in fact the previous reading had been taken with the Print Roll *where it is now* with respect to the web image, the error *would have been* a +5, *not* -40. So, for the purposes of averaging the error in *this* repeat, we adjust the error of the *last* repeat to correspond to *this* repeat, and average 0 an +5, arriving at a much more satisfying averaged error of +2.5, resulting in a much more benign -2.5 correction.

If we're averaging more than two error readings, the same reasoning applies to the readings previous to the current and last reading, only it's a bit more complicated. Say if the current reading is the n-th reading, then the last reading is the n-1 reading, and the one before it is the n-2 reading, and so on. Now we've just argued that the n-1 reading must be adjusted and corrected to the n-th (current) reading. The n-2 reading must also be adjusted to the n-th (current) reading, but before doing that, it must be adjusted to the n-1 reading. Similarly, the n-3 reading must be adjusted to the n-2 reading, then to the n-1 reading, and finally to the n-th (current) reading, and so on and so on for each reading deeper in the history.

Fortunately, there is a shortcut in actual practice. As stated above, the Registration Controller applies the motor movement correction to the last n raw readings in the moving average array as soon as it has calculated the average error and started the motor movement, by adding the motor movement correction to each element in the array. That renders the moving average array useless for re-calculating the current average error, but it doesn't need to do that again, and it "sets up" the array for calculating the average error on the event of the next repeat.

What happens when the controller "dumps" the moving average array? Earlier I made reference to a value representing the number of "participating" registration errors to be averaged, which is always less than or equal to the operator selected number of repeats to average. If the Registration Controller sets this value to zero, that indicates that there are no entries, and therefore no past history, in the moving average array. When the press stops, or when the Registration Controller reads a "large value," the Registration Controller sets the number of repeats in the history to zero, effectively "dumping" the array.

The reason for "dumping" the array is this: if the press stops, or if we read scandalously large error values, then all of the past history is suspect and probably of no worth. Better to start over and build a new array.

Effectively what happens when we build into an empty is this: The Registration Controller reads the first Registration Error and places it into the moving average array, and, since it's the only entry, it's effectively not averaged. Then the Registration Controller reads the second Registration Error and places it into the moving average array after the first, and averages the second with the first. After that, if averaging more than two, the Registration Controller reads the third Registration Error and places it into the moving average array after the second, and averages the third with second and first. If there are more to be averaged, the

Registration Controller repeats this process until the specified number of errors has been reached. After that, the oldest error reading is discarded when a new Registration Error is entered into the array.

1. Windowing. The default behavior of the Registration System is to assume that the Web registration mark lies in a "clear track." That is, the only thing that interrupts the web photo optic sensor's light beam should be the registration mark(s) — no image, no stray ink, no stray marks. Alternatively, the operator may dial in an "inspection zone" that forms a "window" on the Web, within which the Registration Controller looks for the registration mark. The Registration Controller "sees," but ignores photo sensor triggers outside of "inspection zone."

1. Wander Alarms.

1. Variable Repeat

Set-up and maintenance

1. Set Station Address

1. Set Machine Model Number,

1. Set Operating Parms

1. Select Print Roll Drive Side

Pre-registration

1.