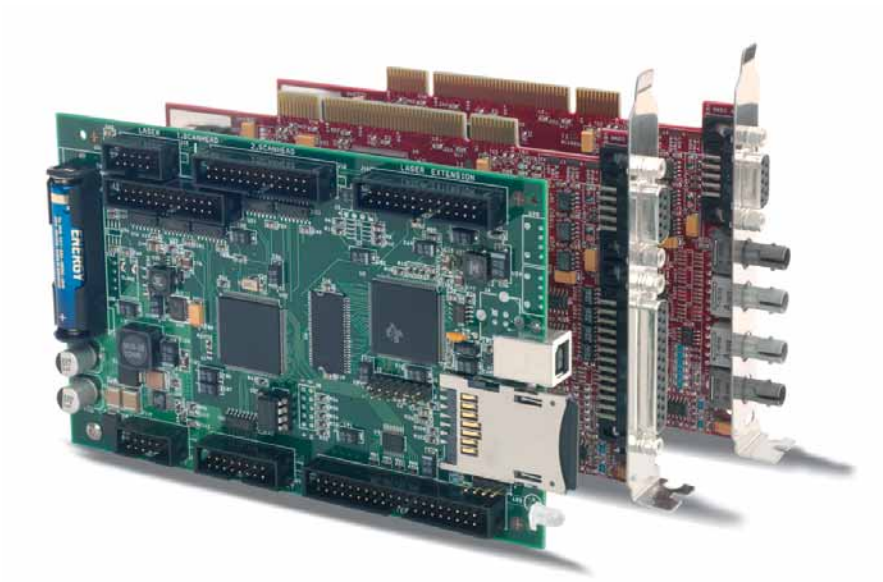
SCANLAB
innovators for industry

Installation and Operation

Processing-On-The-Fly Software

for the RTC®3, RTC®4 and RTC® *SCAN*alone boards



SCANLAB AG
Siemensstr. 2a
82178 Puchheim
Germany

Tel. +49 (89) 800 746-0
Fax: +49 (89) 800 746-199

info@scanlab.de
www.scanlab.de

(Doc. Rev. 1.7 e - March 5, 2010)

SCANLAB
innovators for industry

Contents

1 Delivered Package

1.1 Package Contents

The Processing-On-The-Fly Software software is supplied on the same diskette that contains the standard RTC®3, RTC®4 or RTC® SCANalone software.

1.2 Manufacturer

SCANLAB AG
Siemensstr. 2a
82178 Puchheim
Germany

Tel. +49 (89) 800 746-0
Fax: +49 (89) 800 746-199

info@scanlab.de
www.scanlab.de

1.3 About This Operating Manual

This operating manual is a part of the product. Please read these instructions carefully before you proceed with connecting cables to the MARKING ON THE FLY connector and using the Processing-On-The-Fly Software software. If there are any questions regarding the contents of this manual, please contact SCANLAB.

Keep the manual available for servicing, repairs and disposal. This manual should accompany the product if ownership changes hands.

This manual refers to the following versions of the RTC®3 software and firmware:

- DLL: `RTC3DLL.DLL`, version 148 or higher
- DSP program files: `RTC3D2.HEX` / `RTC3D3.HEX`, version 2.066 / 3.066 or higher
- RTC®3 firmware: version 58 or higher

and to the following versions of the RTC®4 software and firmware:

- DLL: `RTC4DLL.DLL`, version 400 or higher
- DSP program files: `RTC4D2.HEX` / `RTC4D3.HEX`, version 2.400 / 3.400 or higher
- RTC®4 firmware: version 128 or higher

and to the following versions of the RTC® SCANalone software and firmware:

- DLL: `RTC_SCANalone.DLL`, version 500 or higher
- DSP program: (Hex-)version 2.500 / 3.500 or higher
- RTC® SCANalone firmware: version 128 or higher

2 Processing-On-The-Fly – Principle of Operation

2.1 Intended Use

The Processing-On-The-Fly option for SCANLAB's SCANLAB's RTC®3 or RTC®4 PC interface board or SCANLAB's RTC® SCANalone standalone board enables processing of parts in motion, e.g. parts on a conveyor belt or on a rotating plate.

The vectors transferred to the scan head are modified by the RTC® board in accordance with the movement of the parts to be processed. This is described in detail in the following sections.

Focusing a laser beam on a flat surface is possible if the scan head is equipped with an F-Theta lens. The Processing-On-The-Fly option can also be used in combination with a vario*SCAN* dynamic focusing unit from SCANLAB.

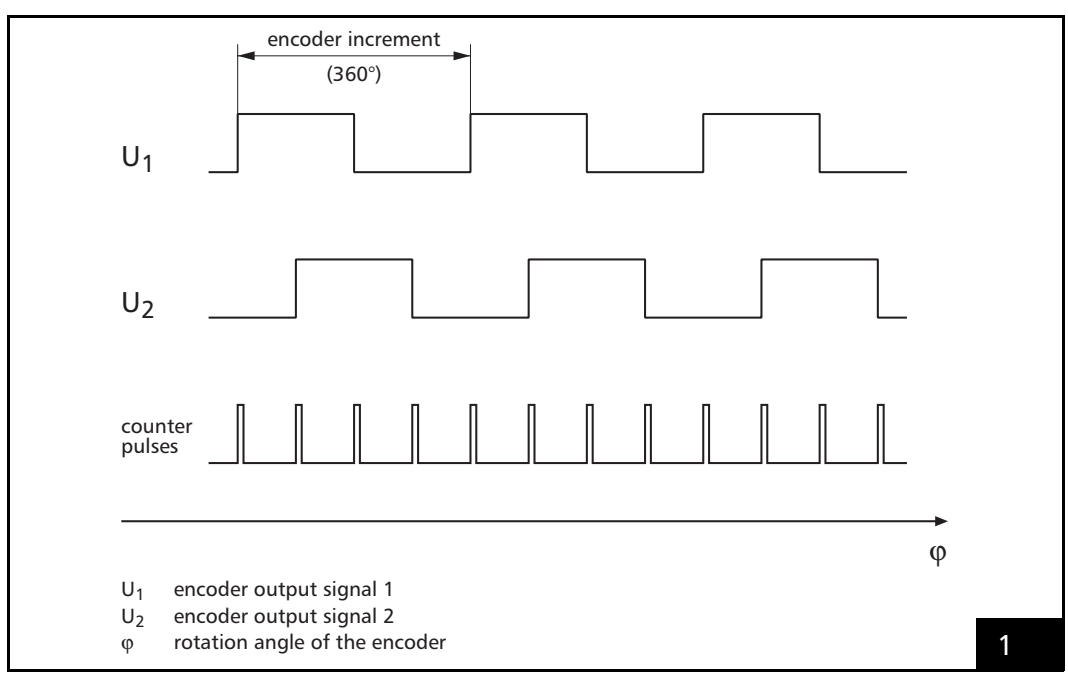
2.2 Detecting the Object's Movement

Inputs for Encoder Signals

For taking account of the varying speed of moving parts, the RTC® board can evaluate the signals of a user-supplied incremental encoder, which might be coupled to a conveyor belt or to a rotating plate. The encoder signals are fed into a 32-bit counter on the RTC® board. Each encoder input is designed for a pair of standardized differential input signals (RS-422).

Figure 1 shows the timing diagram of typical encoder signals. The second encoder signal is usually phase-shifted by 90°. The RTC® counter triggers at each edge of both signals, i.e. one encoder increment results in four counter pulses (counts).

The 90° phase-shift of the second encoder signal allows the RTC® board to detect not only the speed but the direction of movement as well. Depending on the direction of movement, the counter value (signed 32-bit) is increased or decreased.



Timing diagram of typical encoder signals and of the corresponding RTC® counter pulses

For linear movements of the parts to be processed the RTC® board provides two separate encoder inputs: ENCODER X and ENCODER Y. Thus two independent perpendicular movements can be detected.

For rotational movements only one encoder input is necessary. Here ENCODER X must be used.

The input signals of the user-supplied encoder must not excced a maximum allowed frequency of 2MHz.

Encoder Simulation

If the object moves at a constant speed, an encoder is often not necessary. A periodic clock signal can be used instead.

The RTC® provides a 1 MHz clock signal for this purpose. See the command **simulate_encoder** on page 15 for details.

2.3 Movement Compensation During Marking

While the object is moving, all vector data transferred to the scan head must be corrected, depending on the object's current position. The correction is started via list commands. The corresponding calibration parameters have to be pre-determined.

On an RTC® SCANalone board, control commands can only be used in PC operation (where they can also be used for initializing the RTC® SCANalone in stand-alone operation). Therefore, pre-determination of calibration parameters can only be performed in PC operation.

Linear Movements

The correction for a linear movement is started via the list commands **set_fly_x** or **set_fly_y**. A scaling factor [in bits per count] must thereby be specified, which defines the shift [in bits] of the current output position in the image field(1) corresponding to one counter pulse (see below).

When the command **set_fly_x** (or **set_fly_y**) is called, the counter is set to zero. Thereafter, the product of the scaling factor and the current counter value is added to the current output position. This correction is performed every 10 µs.

(1) For a definition of the image field coordinate system, please refer to the RTC®3, RTC®4 or RTC® SCANalone manual.

The Processing-On-The-Fly correction can be stopped (simultaneously for both directions) via the command **fly_return**. The new output position can be thereby defined.

Scaling factor

If an encoder is used to detect the Processing-On-The-Fly speed of the parts to be processed, the encoder must be calibrated before the Processing-On-The-Fly software compensation can be used.

First the command **get_encoder** must be used to determine the encoder increment i_x and i_y [in counts per mm] for each direction:

- Read the counter start value (via **get_encoder**) and begin the movement
- Stop the movement and read the counter end value (via **get_encoder**)
- Measure the distance travelled in mm
- Encoder increment i = (counter end value - counter start value) / distance travelled

In a second step calculate the scaling factors k_x and k_y [in bits per count] from the encoder increments and the scan system's calibration factor K [in bits per mm] as specified in its operating manual:

$$k_x = K / i_x$$
$$k_y = K / i_y$$

If the object moves at a constant speed v_x or v_y [in mm per second] and the encoder simulation is used instead of an encoder, then the scaling factor k_x and k_y are calculated as follows:

$$k_x = K \cdot v_x / (1{,}000{,}000 \text{ counts/second})$$
$$k_y = K \cdot v_y / (1{,}000{,}000 \text{ counts/second})$$

It might be necessary to correct the implied direction of movement by inverting the sign of k_x and k_y.

Rotary Movements

Before starting the correction for a rotary movement, the rotation center of the Processing-On-The-Fly rotation must be defined. Therefore the list command **set_rot_center** is provided. The rotation center may also be situated outside the image field.

The correction itself is started via the list command **set_fly_rot**. The encoder `resolution` [in counts per revolution] must thereby be specified. The inverse of the encoder `resolution` defines the rotation [in number of revolutions] of the current output position in the image field corresponding to one counter pulse (see below).

When the command **set_fly_rot** is called, the counter is set to zero. The output value is then calculated from the current output position, the encoder `resolution` and the current counter value. This correction is performed every 10 µs.

To stop Processing-On-The-Fly correction, the command **fly_return** should be used. The new output position can be thereby defined.

Encoder Resolution

If an encoder is used to detect the rotating speed of the parts to be processed, the encoder must be calibrated before the Processing-On-The-Fly software compensation can be used.

The command **get_encoder** must be used to determine the encoder `resolution` [in counts per revolution]:

- read the counter start value (via **get_encoder**) and begin the rotation (a certain number of full rotations)
- count the number of revolutions
- stop the rotation and read the counter end value (via **get_encoder**)
- `resolution` = (counter end value - counter start value) / number of revolutions

Note that the number of counts per revolution is four times the number of encoder pulses per revolution (see page 5).

If the object rotates at a constant speed ω[in number of revolutions per second] and the encoder simulation is used instead of an encoder, then the encoder `resolution` is calculated as follows:

`resolution` = (1,000,000 counts/second) / ω

It might be necessary to correct the implied direction of movement by inverting the sign of the encoder `resolution`.

2.4 Synchronization (Start of Marking)

The transfer of marking commands from the RTC® list buffer to the scan head can be started by either a software command or an external start signal (see the RTC®3, RTC®4 or RTC® SCANalone manual).

The MARKING ON THE FLY connector on the RTC® board provides additional START and STOP inputs (see the RTC®3, RTC®4 or RTC® SCANalone manual).

It is possible, for example, to connect the external start signal to a light barrier or a mechanical contact.

A delay (i.e. a number of encoder pulses) between the START pulse and the actual start of the marking routine can be programmed with the command **set_ext_start_delay** (see page 12) or **set_ext_start_delay_list** (see page 12).

To execute several lists with defined spacings, the list command **simulate_ext_start** (see page 16) can be used.

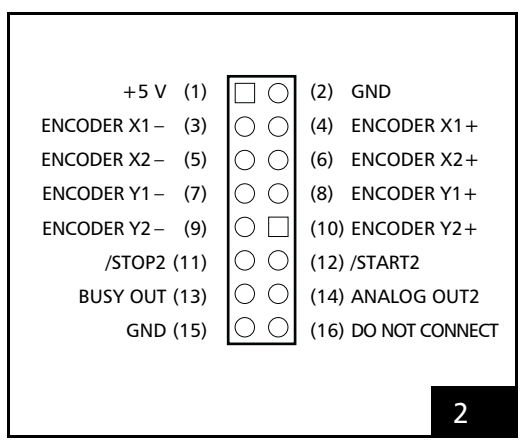
2.5 Monitoring Processing-On-The-Fly Overflows

For monitoring Processing-On-The-Fly overflows the command **get_marking_info** on page 11 is provided.
Processing-On-The-Fly overflows can occur, if a Processing-On-The-Fly application program is not optimized for the intended Processing-On-The-Fly speed or if an accidental considerable increase of the Processing-On-The-Fly speed occurs during operation. In this case the scan system's scanners reach their extreme positions and the scan system will not mark the intended pattern. Regularly calling the **get_marking_info** command during test runs or during normal operation helps to avoid failures caused by Processing-On-The-Fly overflows.

3 Installation

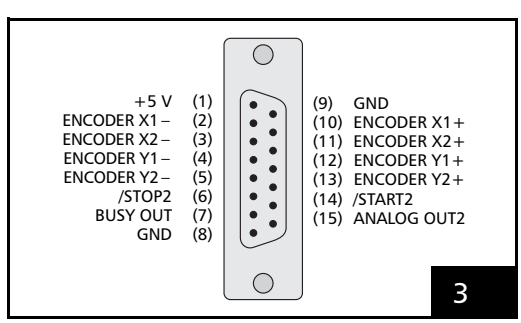
3.1 Electrical Connections

Figure 2 shows the pin configuration of the MARKING ON THE FLY connector on the RTC® board. The connector is located near the top of the board – see the RTC®3, RTC®4 or RTC® SCANalone manual.



Pin-out of the on-board MARKING ON THE FLY connector

SCANLAB recommends using an additional PC slot cover for using the inputs and signals of the MARKING ON THE FLY connector. A suitable slot cover with a 15-pin female D-SUB connector is available from SCANLAB. Figure 3 shows the pin-out of this D-SUB connector.



Pin-out of the 15-pin female D-SUB connector of the optional MARKING ON THE FLY slot cover

Encoder Inputs

Each of the encoder inputs ENCODER X and ENCODER Y are designed for a pair of standardized RS-422 differential signals (see figure 1 on page 5).

The HIGH level input voltage should be at least 2.5 V, the LOW level voltage at maximum 0.5 V.

DC Voltage Output

At pin (1) a DC voltage output of (+5 ± 0.25) V referenced to GND is provided as a power supply for the encoders.

The maximum current load is 100 mA.

External Control Inputs

The signal inputs /START2 and /STOP2 (TTL active-low) are similar to the /START and /STOP inputs at the 9-pin laser connector of the RTC®.
Please refer to the RTC®3, RTC®4 or RTC® SCANalone manual for details.

To start a list, either the /START signal or the /START2 signal can be used. Likewise, execution will be aborted if either the /STOP signal or the /STOP2 signal becomes LOW.

The START inputs must be enabled by the command **set_control_mode** (see the RTC®3, RTC®4 or RTC® SCANalone manual).

Output Signals

ANALOG OUT2

The 10-bit ANALOG OUT2 output port can be used for any purpose. To load that port with a desired value, use the commands **write_da_2** or **write_da_2_list** (see the RTC®3, RTC®4 or RTC® SCANalone manual).

The voltage range of the ANALOG OUT2 port is 0 V...+10 V. The maximum current load is 5 mA.

BUSY OUT (Status Signal, TTL Level)

The BUSY OUT signal is HIGH when a list is currently executing – see also **get_status** (see the RTC®3, RTC®4 or RTC® SCANalone manual).

The maximum current load is 10 mA.

3.2 Software

The standard RTC® driver DLL (`RTC3DLL.DLL`, `RTC4DLL.DLL` or `RTC_SCANalone.DLL`) includes all necessary commands for Processing-On-The-Fly applications. No additional software is needed.

To use the RTC® with the Processing-On-The-Fly option, the MARKING ON THE FLY connector on the RTC® board must be enabled. Please contact SCANLAB for details.

To check whether the Processing-On-The-Fly option is installed, you can use the command **get_rtc_version** (see the RTC®3, RTC®4 or RTC® SCANalone manual).

4 Software Commands

4.1 Command Set

The commands are listed in alphabetical order.

In combination with an RTC®3 or RTC®4 board, all Processing-On-The-Fly commands can also be used as multi-board commands (see the RTC®3 or RTC®4 manual).

List Command	**fly_return**	
Function	turns off the Processing-On-The-Fly correction and defines the new output position	
Parameters	`x, y`	new output position values, signed 16-bit values
Integration	Pascal:	`procedure fly_return(var x, y: smallint);`
	C:	`void fly_return(short *x, short *y);`
	Basic:	`sub fly_return(x%, y%)`
Comments	• The **fly_return** command simultaneously turns off the Processing-On-The-Fly correction started with **set_fly_x** and **set_fly_y**. It also turns off the Processing-On-The-Fly correction started with **set_fly_rot**. • Directly after the Processing-On-The-Fly is turned off, a jump to the defined position is executed.	
References	**set_fly_x**, **set_fly_y**, **set_fly_rot**	

Ctrl Command	**get_encoder**	
Function	returns the encoder counter values (lower 16 bits)	
Result	`zx, zy`	lower 16 bits of the encoder counter values as signed 16-bit values
Integration	Pascal:	`procedure get_encoder(var zx, zy: smallint);`
	C:	`void get_encoder(short *zx, short *zy);`
	Basic:	`sub get_encoder(zx%, zy%)`
Comments	• On an RTC® SCANalone board, this command can only be used in PC operation. • This command is mainly needed for determining the encoder increment [in counts per millimeter] and therefore for calibrating the scaling factors `kx` and `ky` (see page 6) or for determining the encoder `resolution` [in counts per revolution] (see page 7). • The upper 16 bits of the counter values are not returned.	
References	**set_fly_x**, **set_fly_y**, **set_fly_rot**	

Ctrl Command	get_marking_info
Function	returns information about errors occurred since the last call of the **get_marking_info** command (or since the last system start-up)
Result	Error signal word (unsigned 16-bit value): Bit #0 (LSB) = 1: Processing-On-The-Fly overflow at the left border of the image field (X < –32768) Bit #1 = 1: Processing-On-The-Fly overflow at the right border of the image field (X > +32767) Bit #2 = 1: Processing-On-The-Fly overflow at the bottom border of the image field (Y < –32768) Bit #3 = 1: Processing-On-The-Fly overflow at the top border of the image field (Y > +32767) Bit #8 =1: TriggerError: an enabled external trigger or simulated trigger occured during executing a list The remaining bits are reserved.
Integration	Pascal: `function get_marking_info: word;`
	C: `unsigned short get_marking_info(void);`
	Basic: `function get_marking_info()%`
Comments	• **This command can only be used together with** – **an RTC®3, the DLL** `RTC3DLL.DLL` **version 154 (or higher) and the DSP program file** `RTC3D2.HEX` / `RTC3D3.HEX` **version 2.073 / 3.073 (or higher). Bit #8 is not available on RTC®3 boards.** – **an RTC®4, the DLL** `RTC4DLL.DLL` **version 427 (or higher) and the DSP program file** `RTC4D2.HEX` / `RTC4D3.HEX` **version 2.413 / 3.413 (or higher).** – **an RTC® SCANalone, the DLL** `RTC_SCANalone.DLL` **version 517 (or higher) and the DSP program (Hex-version) 2.513 / 3.513 or higher.** • All error bits are reset during a start-up and after the command **get_marking_info** is executed.

Ctrl Command	set_ext_start_delay	
Function	defines a delay (counter pulses) to be inserted after each external START signal (before the following list is started)	
Parameters	`delay`	signed 16-bit value; see comments
	`encoder`	= 0: use Encoder X = 1: use Encoder Y
Integration	Pascal:	`procedure set_ext_start_delay(delay, encoder: smallint);`
	C:	`void set_ext_start_delay(short delay, short encoder);`
	Basic:	`sub set_ext_start_delay(ByVal delay%, ByVal encoder%)`
Comments	• On an RTC® SCANalone board, this command can only be used in PC operation (where it can also be used for initializing the RTC® SCANalone in stand-alone operation). • This command can be only used together with a real encoder (hardware encoder), *not* together with **simulate_encoder**. • A separate counter is used for the external start delay. When the RTC® receives an external start signal, the counter is set to zero. Afterwards the pulses from the selected encoder are counted. When the counter reaches the value (`delay` • 16), the list is started. (The list is started only if no list is executing at the moment.) • **Note:** The encoder pulses are counted *with sign*, i.e. depending on the direction of the movement. If the object moves in a negative direction, the `delay` parameter should be negative as well. • Use the command **set_extstartpos_list** or **select_list** (see the RTC®3, RTC®4 or RTC® SCANalone manual) to specify the start position of the list to be executed. • Setting the parameter `delay` to zero turns off the external start delay.	
References	**simulate_ext_start**, **set_ext_start_delay_list** **select_list**, **set_extstartpos_list**, **set_control_mode** (see the RTC®3, RTC®4 or RTC® SCANalone manual)	

List Command	set_ext_start_delay_list	
Function	same as **set_ext_start_delay**, but a list command	
Integration	Pascal:	`procedure set_ext_start_delay_list(delay, encoder: smallint);`
	C:	`void set_ext_start_delay_list(short delay, short encoder);`
	Basic:	`sub set_ext_start_delay_list(ByVal delay%, ByVal encoder%)`
Comments	• On an RTC® SCANalone board, this command can be used in PC operation *and* in stand-alone operation.	

List Command	set_fly_rot
Function	sets the encoder `resolution` for Encoder X and starts Processing-On-The-Fly for compensation of a rotary movement
Parameter	`resolution` encoder pulses per revolution, 64-bit IEEE floating point format
Integration	Pascal: `procedure set_fly_rot(var resolution: double);`
	C: `void set_fly_rot(double resolution);`
	Basic: `sub set_fly_rot(ByVal resolution#)`
Comments	• When the **set_fly_rot** command is executed, the ENCODER X counter resets to zero. • After this command executes, the X and Y value of the current output position are continuously corrected with respect to the current ENCODER X counter value. This correction is performed every 10 µs. • Before executing the command **set_fly_rot**, the rotation center must be defined via the command **set_rot_center**. • Use the **get_encoder** command to calibrate the encoder, i.e. to determine the encoder `resolution` [in counts per revolution]. Note, that the number of counts per revolution is four times the number of encoder pulses per revolution (see page 5). It might be necessary to correct the implied direction of movement by inverting the sign of the encoder `resolution`. • The encoder counters are signed 32-bit counters. Overflows will not be intercepted. • Use the **fly_return** command to turn off Processing-On-The-Fly correction and thereby define a new output position. • If the absolute value of the encoder `resolution` is smaller than 100, then the Processing-On-The-Fly correction is also turned off. The end point of the last transferred vector is then repeated without the correction.
References	**set_rot_center**, **get_encoder**, **fly_return**

Ctrl Command	set_rot_center
Function	sets the rotation center of a Processing-On-The-Fly rotation
Parameter	`center_x`, `center_y` position values of the rotation center, signed 32-bit value
Integration	Pascal: `procedure set_rot_center(center_x, center_y: longint);`
	C: `void set_rot_center(long center_x, long center_y);`
	Basic: `sub set_rot_center(ByVal center_x As long, ByVal center_y As long)`
Comments	• On an RTC® SCANalone board, this command can only be used in PC operation (where it can also be used for initializing the RTC® SCANalone in stand-alone operation). • The **set_rot_center** command should be executed before the Processing-On-The-Fly rotation is started via **set_fly_rot**. • The center position values are referenced to the zero point (0|0). The absolute values must not exceed 8,000,000 • The rotation center may be situated outside the image field.
References	**set_fly_rot**

List Command	set_fly_x	
Function	sets the scaling factor for Encoder X and starts Processing-On-The-Fly for compensation of a linear movement in the X direction	
Parameter	kx	scaling factor [bits/count], 64-bit IEEE floating point format
Integration	Pascal:	`procedure set_fly_x(var kx: double);`
	C:	`void set_fly_x(double kx);`
	Basic:	`sub set_fly_x(ByVal kx#)`
Comments	• When the **set_fly_x** command is executed, the ENCODER X counter resets to zero. • After this command executes, the X value of the current output position is continuously corrected with respect to the current ENCODER X counter value. This correction is performed every 10 µs. • The encoder counters are signed 32-bit counters. Overflows will not be intercepted. • Use the **get_encoder** command to calibrate the encoder, i.e. to determine the encoder increment [in counts per millimeter]. • The scaling factor must be specified in *bits per count*. For calculating the scaling factor see page 6. It might be necessary to correct the implied direction of movement by inverting the sign of the scaling factor. • Use the **fly_return** command to turn off Processing-On-The-Fly correction and thereby define a new output position. • If the scaling factor is set to zero, then the Processing-On-The-Fly correction is also turned off. The end point of the last transferred vector is then repeated without the correction.	
References	**get_encoder**, **fly_return**	

List Command	set_fly_y	
Function	sets the scaling factor for Encoder Y and starts Processing-On-The-Fly for compensation of a linear movement in the Y direction	
Parameter	ky	scaling factor [bits/count], 64-bit IEEE floating point format
Integration	Pascal:	`procedure set_fly_y(var ky: double);`
	C:	`void set_fly_y(double ky);`
	Basic:	`sub set_fly_y(ByVal ky#)`
Comments	• When the **set_fly_y** command executes, the ENCODER Y counter resets to zero. • After this command executes, the Y value of the current output position is continuously corrected with respect to the current ENCODER Y counter value. This correction is performed every 10 µs. • The encoder counters are signed 32-bit counters. Overflows will not be intercepted. • Use the **get_encoder** command to calibrate the encoder, i.e. to determine the encoder increment [in counts per millimeter]. • The scaling factor must be specified in *bits per count*. For calculating the scaling factor see page 6. It might be necessary to correct the implied direction of movement by inverting the sign of the scaling factor. • Use the **fly_return** command to turn off Processing-On-The-Fly correction and thereby define a new output position. • If the scaling factor is set to zero, then the Processing-On-The-Fly correction is also turned off. The end point of the last transferred vector is then repeated without the correction.	
References	**get_encoder**, **fly_return**	

Ctrl Command	simulate_encoder	
Function	simulates ENCODER X or ENCODER Y pulses with a 1 MHz clock signal	
Parameter	`channel`	= 1: simulates the ENCODER X pulses = 2: simulates the ENCODER Y pulses = 3: simulates both the ENCODER X and the ENCODER Y pulses = 0: off
Integration	Pascal:	`procedure simulate_encoder(var channel: word);`
	C:	`void simulate_encoder(unsigned short channel);`
	Basic:	`sub simulate_encoder(ByVal channel%)`
Comments	• On an RTC® SCANalone board, this command can only be used in PC operation (where it can also be used for initializing the RTC® SCANalone in stand-alone operation). • This command can be used, for example, in combination with a conveyor belt or a rotating plate moving at a constant speed. The corresponding encoder counter is increased at a time rate of 1 MHz. • To calibrate the Processing-On-The-Fly speed and direction, the command **set_fly_x**, **set_fly_y** or **set_fly_rot** must be used (see page 6 or page 7).	
References	**set_fly_x**, **set_fly_y**, **set_fly_rot**	

List Command	simulate_ext_start
Function	starts the next list automatically after a specified delay (counter pulses)
Parameters	`delay` signed 16-bit value; see comments `encoder` = 0: use encoder X = 1: use encoder Y
Integration	Pascal: `procedure simulate_ext_start(delay, encoder: smallint);` C: `void simulate_ext_start(short delay, short encoder);` Basic: `sub simulate_ext_start(ByVal delay%, ByVal encoder%)`
Comments	• This command is a list command. It should be used as the *first* command in a list. • A separate counter is used for the start delay. When the command **simulate_ext_start** is executed, the RTC® sets the counter to zero. Afterwards, the pulses from the selected encoder are counted. When the counter reaches the value (`delay` • 16), the external start is performed. (The next list is started only if no list is executing at the moment, i.e. if the current list is already finished.) • **Note:** The counter pulses are counted *with sign*, i.e. depending on the direction of the movement. If the object moves in the reverse direction, the parameter `delay` should be negative as well. • Use the command **set_extstartpos_list** (see the RTC®3, RTC®4 or RTC® SCANalone manual) to specify the start position of the next list. • With an RTC®4 PC interface board (only with DSP program files `RTC4D2.HEX` / `RTC4D3.HEX` version 2.433 / 3.433 or higher and a DLL file `RTC4DLL.DLL` version 440 or higher), `encoder` = 10 (as `encoder` = 0) can be used to select encoder X and `encoder` = 11 (as `encoder` = 1) to select encoder Y, too. With `encoder` = 10 or 11 (in contrast to `encoder` = 0 or 1), the list start is only executed, if the external /START (or /START2) signal is active, i.e. if this signal has been enabled with **set_control_mode** and activated via a HIGH to LOW level transition. In addition, with `encoder` = 10 or 11 (in contrast to `encoder` = 0 or 1), the simulated start is ignored, when the maximum number of external list starts (defined via **set_max_counts**) has been reached.
References	**set_ext_start_delay**, **set_ext_start_delay_list** **select_list**, **set_control_mode** (see the RTC®3, RTC®4 or RTC® SCANalone manual)

4.2 Emulated RTC®2 Commands

The following command was implemented in the RTC®3 software for compatibility (*not* included in the RTC®4 or RTC® SCANalone software). For programming new applications, however, only the commands described in chapter 4.1 should be used.

List Command	**set_encoder_values**	
Function	sets the encoder scaling factors kx and ky	
Parameters	kx, ky	scaling factors [bits/count], 64-bit IEEE floating point format
	z1, z2	ignored
Integration	Pascal:	`procedure set_encoder_values(kx, ky: double; z1, z2: smallint);`
	C:	`void set_encoder_values(double kx, double ky, short z1, short z2);`
	Basic:	`sub set_encoder_values(ByVal kx#, ByVal ky#, ByVal z1%, ByVal z2%)`
Comments	• The encoder counters are automatically set to zero. • The parameters z1 and z2 are ignored. • This command calls the commands **set_fly_x** and **set_fly_y**.	
References	**set_fly_x**, **set_fly_y**	

5 Technical Specifications

Encoder Inputs

U_{High}	≥ 2.5 V
U_{Low}	≤ 0.5 V
f	≤ 2MHz

DC Voltage Output

Voltage	(+5 ± 0.25) V DC
Maximum current load	100 mA

ANALOG OUT2 Port

Voltage Range	0 V ...+10 V
Maximum current load	5 mA

BUSY OUT Signal

Voltage Range	TTL level
Maximum current load	10 mA

Index

T

V

SCANLAB
innovators for industry

Notes