Chapter 15

PROGRAMMER

Edition 3

Overview

This class of function blocks contains both simple ramp function blocks which can be used to ramp the value of a variable in a linear, time dependent fashion and more complex program function blocks which allow multi-segment ramp/dwell profiles to be created. Both function blocks are available in ramp rate and time to target versions.

RAMP_TARGET FUNCTION BLOCK

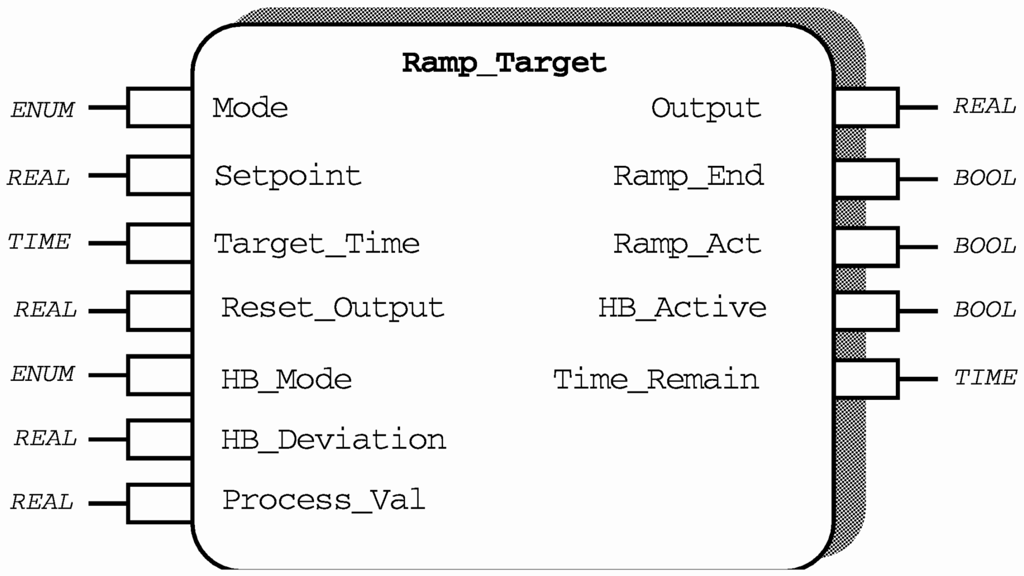


Figure 15-1 Ramp_Target Function Block Diagram

Functional Description

The Ramp_Target function block ramps the **Output** towards a target **Setpoint** so that this setpoint is achieved in the time given by **Target_Time.** The block has three modes of operation, which are defined by the **Mode**. **Holdback** functionality can be activated to restrict the **Rate** in the event of a sluggish **Process_Val.**

Holdback serves to hold the **Output** at a constant value in the event that the deviation between the **Output** and the **Process_Val** exceeds the value defined by **HB_Deviation**. The Figure below shows holdback acting in response to the **Process_Val** lagging behind the increasing ramp **Output.** When the **Process_Val** input deviates from the ramp **Output** by more than **HB_Deviation,** the **Output** is held at constant value until the deviation decreases below **HB_Deviation.** In the case shown, this has the effect of increasing the effective time to achieve setpoint above that indicated by **Target_Time**.

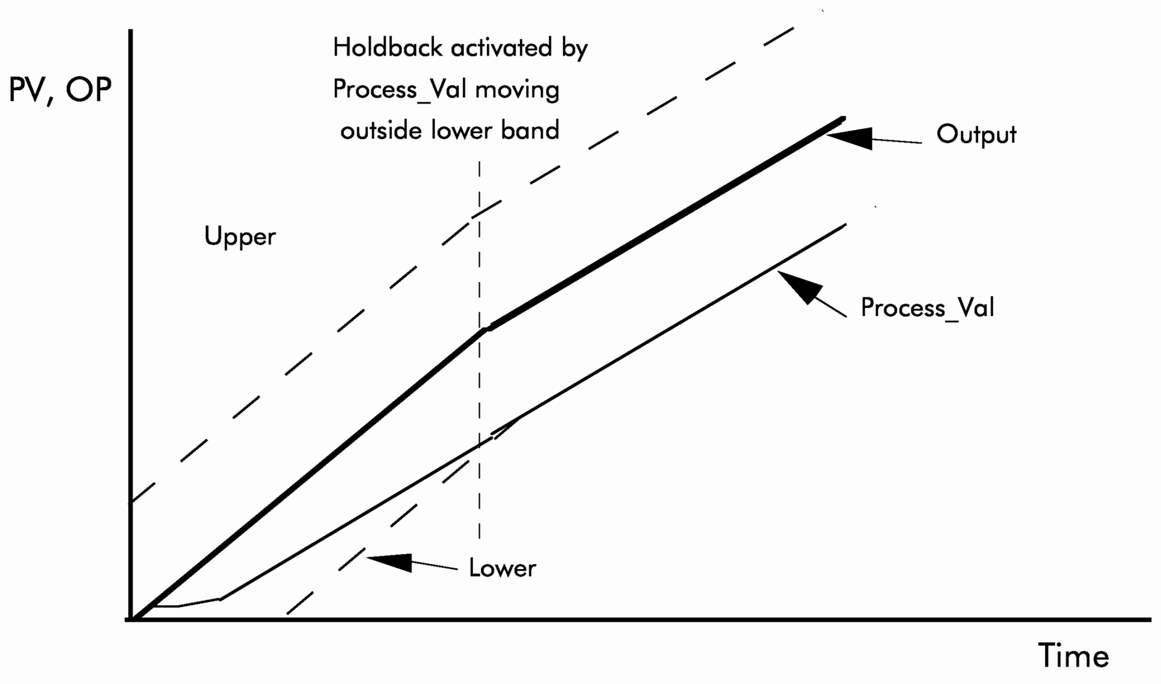


Figure 15-2 Holdback operation with lagging Process_Val

Function Block Attributes

Type:	3F03
Class:	PROGRAMMER
Default Task:	Task_2
Short List:	Mode, Setpoint, Target_Time, Output
Memory Requirements:	60 Bytes

Parameter Descriptions

Mode (M)

Determines whether **Output** is being updated or not.

Reset(0): in Reset mode the Output will be track **Reset_Output** and **Ramp_Act** will be set to No.

Run(1): in Run mode the **Output** will ramp towards **Setpoint** at a constant rate such that setpoint will be achieved in **Target_Time** provided the block remains in Run mode.

Hold(2): in Hold mode the **Output** will remain constant at the value reached before entering Hold mode.

Setpoint (SP)

The **Setpoint** is the target value to which the **Output** is ramped.

Target_Time (TT)

The time that is to be taken in changing **Output** from its current value to **Setpoint.** When the block is set into Run mode from Reset, the rate required to achieve this change in output is calculated and maintained unless some change occurs to one of the function block inputs.

If holdback becomes active or Mode is switched to Hold, the **Output** is frozen. When holdback is no longer active or mode reverts to Run, the output continues changing at the calculated rate towards output provided neither **Target_Time** nor **Setpoint** have been changed during the period of hold.

If a change is made to **Target_Time** or Setpoint whilst the function block is in Run mode, the rate of change is immediately recalculated and the **Output** starts changing at this new rate.

Reset_Output (ROP)

The value of this input drives **Output** whilst the function block is in Reset mode.

HB_Mode (HBM)

Determines the way in which holdback operates.

Off(0): holdback is disabled.

Lower(1): holdback will be enabled when the Output minus **Process_Val** is greater than **HB_Deviation.**

Upper(2): holdback will be enabled when the **Process_Val** minus **Output** is greater than the **HB_Deviation.**

Band(3): holdback will be enabled when the absolute value of **Output** minus **Process_Val** is greater than the **HB_Deviation.**

HB_Deviation (HBD)

The **HB_Deviation** parameter defines the amount of deviation allowed between Output and **Process_Val** before holdback is applied, subject to **HB_Mode**.

Process_Val (PV)

The **Process_Val** parameter operates in conjunction with **HB_Deviation** to determine whether holdback is active. The parameter is not used if **HB_Mode** is set to Off.

Output (OP)

The **Output** parameter is the Real output of the Ramp Target function block, which will ramp towards **Setpoint** when the **Mode** is set to Run and the function block is not in holdback. The **Output** will equal **Reset_Output** when the **Mode** is set to Reset.

Ramp_End (RE)

The **Ramp_End** parameter defines when the **Output** has completed its ramping to the **Setpoint.** When the **Mode** is equal to Reset, **Ramp_End** will be set to False unless **Setpoint** is equal to **Output.** When the **Mode** is equal to Run, **Ramp_End** will be set to True only when the **Output** equals the **Setpoint.** If the **Setpoint** is changed after **Ramp_End** has become True the **Ramp_End** will then change to False until **Output** is equal to **Setpoint** again. When in holdback, the operation of **Ramp_End** is unchanged, with **Ramp_End** being set to False unless **Output** equals **Setpoint.**

Ramp_Act (RAC)

The **Ramp_Act** defines whether the **Output** is ramping towards **Setpoint.** When the Mode is set to Reset, **Ramp_Act** will be set to No. When the **Mode** is set to Run or Hold, **Ramp_Act** will be set to Yes when the **Output** does not equal **Setpoint** and will be set to No when the **Output** equals the **Setpoint.**

HB_Active (HBA)

The **HB_Active** parameter is an indicator to the action of the holdback function. When the **Mode** parameter is set to Reset (0) or Hold (2), **HB_Active** will equal No (0). When the **Mode** parameter is set to Run (1) and **HB_Mode** is set to Lower (1), Upper (2) or Band (3), **HB_Active** will be equal Yes (1) if the block is in holdback.

Time_Remain (TR)

Indicates the time that will currently be required for **Output** to reach **Setpoint.** This parameter is updated every function block execution cycle.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Mode	**ENUM**	Reset (0)	Oper	Oper	See Parameter descriptions	
Setpoint	**REAL**	0	Oper	Oper	High Limit	+3·402823E+38
					Low Limit	-3·402823E+38
Target_Time	**TIME**	0	Oper	Oper	High Limit	23d23h59m59s999ms
					Low Limit	0ms
Reset_Output	**REAL**	0	Oper	Oper	High Limit	+3·402823E+38
					Low Limit	-3·402823E+38
HB_Mode	**ENUM**	Off (0)	Oper	Oper	See Parameter descriptions	
HB_Deviation	**REAL**	0	Oper	Oper	High Limit	+3·402823E+38
					Low Limit	-3·402823E+38
Process_Val	**REAL**	0	Oper	Block	High Limit	+3·402823E+38
					Low Limit	-3·402823E+38
Output	**REAL**	0	Oper	Block	High Limit	+3·402823E+38
					Low Limit	-3·402823E+38
Ramp_End	**BOOL**	False (0)	Oper	Block	Senses	False(0)
						True(1)
Ramo_Act	**BOOL**	No (0)	Oper	Block	Senses	No(0)
						Yes(1)
HB_Active	**BOOL**	No(0)	Oper	Block	Senses	No(0)
						Yes(1)
Time_Remain	**TIME**	0ms	Oper	Block	High Limit	23d23h59m59s999ms
					Low Limit	0ms

Table 15-1 Ramp_Target Parameter Attributes

RAMP_RATE FUNCTION BLOCK

Prior to version 3.00 this function block was located in class OTHERS and was called Ramp.

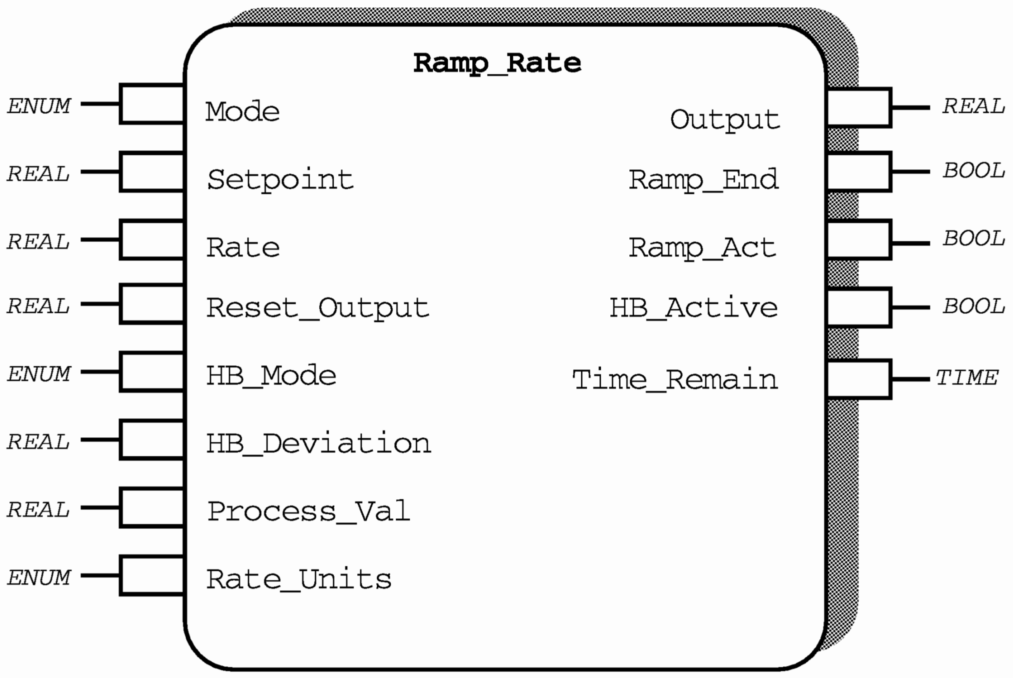


Figure 15-3 Ramp_Rate Function Block

Functional Description

The **Ramp_Rate** function block ramps the **Output** at a constant **Rate** towards a target **Setpoint.** The block has three modes of operation, which are defined by the **Mode**. Holdback functionality can be activated to restrict the **Rate** in the event of a sluggish **Process_Val.**

The remaining time to complete the current ramp is given as an output.

Modes of Operation:

Reset (0):	in Reset mode the Output will be set to **Reset_Output** and **Ramp_Act** will be set to No (0).
Hold (2):	in Hold mode the **Output** will remain constant at the value reached before entering Hold mode
Run (1):	in Run mode the **Output** will ramp towards **Setpoint** at a rate defined by the Rate parameter and Ramp_Act will be set to Yes(1). When the **Output** has reached **Setpoint, Ramp_End** will be set to True (1).

Holdback Operation

Holdback serves to hold the **Output** at a constant value in the event that the deviation between the **Output** and the **Process_Val** exceeds the value defined by **HB_Deviation.** The Figure 15-4 below shows holdback acting in response to the **Process_Val** lagging behind the increasing ramp **Output.** When the **Process_Val** input deviates from the ramp **Output** by more than **HB_Deviation,** the ramping **Output** is held at constant value until the deviation decreases below **HB_Deviation.** In the case shown, this has the effect of limiting the ramp Rate to be equal to the maximum rate of rise of **Process_Val.**

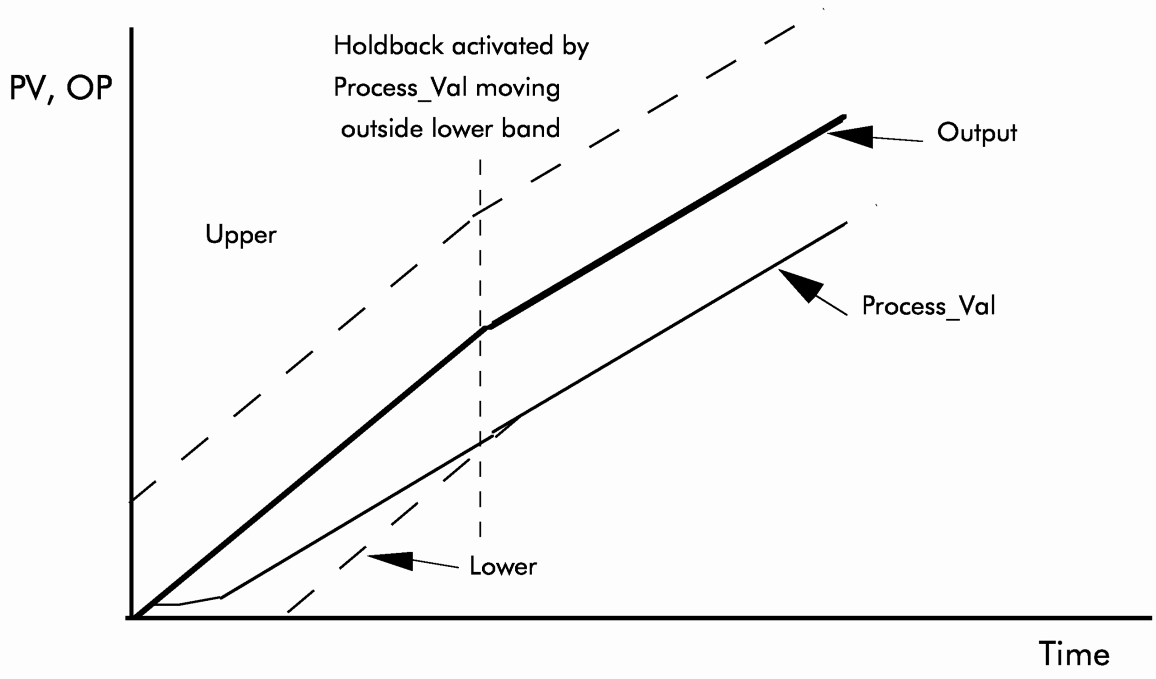


Figure 15-4 Holdback operation with lagging Process_Val.

Modes of Holdback Operation:

OFF(0):	in OFF mode the holdback option is disabled.
LOWER (1):	in LOWER mode the holdback will be enabled when the **Output** minus **Process_Val is** greater than **HB_Deviation.**
UPPER (2):	in UPPER mode the holdback will be enabled when the **Process_Val** minus Output is greater than the **HB_Deviation.**
BAND (3):	in BAND mode the holdback will be enabled when the absolute value of **Output** minus **Process_Val** is greater than the HB_Deviation.

Function Block Attributes

Type:	3F 01
Class:	PROGRAMMER
Default Task:	Task_2
Short List:	Mode, Setpoint, Rate, Output
Memory Requirements:	84 Bytes
Execution Time:	282 µ Secs

Parameter Descriptions

Mode (M)

Mode defines the mode of operation of the function block, see earlier description.

Setpoint (SP)

The **Setpoint is** the target value to which the Output is ramped.

Rate (R)

The **Rate** parameter determines the rate at which the **Output** changes. Its units are defined by the parameter **Rate_Units.**

Reset_Output (ROP)

The **Reset_Output** parameter defines the value written to the parameter **Output** when the **Mode** is Reset.

HB_Mode (HBM)

The **HB_Mode** parameter defines the mode of operation of holdback.

HB_Deviation (HBD)

The **HB_Deviation** parameter defines the amount of deviation allowed between **Output** and **Process_Val** before holdback is applied.

Process_Val (PV)

The **Process_Val** parameter operates in conjunction with **HB_Deviation** to determine whether holdback is active. The parameter is not used if **HB_Mode** is set to Off (0).

Rate_Units (RU)

The **Rate_Units** parameter is used to define the units in which the rate of change of **Output** are defined.

Output (OP)

The **Output** parameter is the Real output of the Ramp_Rate function block, which will ramp towards Setpoint when the **Mode** is set to Run (1) and the function block is not in holdback. The **Output** will equal **Reset_Output** when the **Mode** is set to Reset (0).

Ramp_End (RE)

The **Ramp_End** parameter defines when the **Output** has completed its ramping to the **Setpoint.** When the **Mode** is equal to Reset (0), **Ramp_End** will be set to False (0). When the **Mode** is equal to Run (1), **Ramp_End** will be set to True (1) only when the **Output** equals the **Setpoint.** If the **Setpoint** is changed after **Ramp_End** has become True (1) the **Ramp_End** will then change to False (0) until **Output** is equal to **Setpoint** again. When in holdback, the operation of **Ramp_End** is unchanged, with **Ramp_End** being set to False (0) unless **Output** equals **Setpoint.**

Ramp_Act (RA)

The **Ramp_Act** defines whether the **Output** is ramping towards **Setpoint.** When the **Mode** is set to Reset (0), **Ramp_Act** will be set to No (0). When the **Mode** is set to Run (1) or Hold (2), **Ramp_Act** will be set to Yes (1) when the **Output** does not equal **Setpoint** and will be set to No (0) when the **Output** equals the **Setpoint.**

HB_Active (HBA)

The **HB_Active** parameter is an indicator to the action of the holdback function. When the **Mode** parameter is set to Reset (0) or Hold (2), **HB_Active** will equal No (0). When the **Mode** parameter is set to Run (1) and **HB_Mode is** set to Lower (1), Upper (2) or Band (3), HB_Active will be equal Yes (1) if the block is in holdback.

Time_Remain (TR)

Indicates the time that will be required for the output to reach the setpoint. This parameter is updated at every function block execution.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
HB_Active	**BOOL**	No (0)	Oper	Block	Senses	No (0) Yes (1)
HB_Deviation	**REAL**	0.0	Oper	Oper	High Limit Low Limit	999,999 -99,999
HB_Mode	**ENUM**	Off (0)	Oper	Oper	Senses	Off (0) Lower (1) Upper (2) Band (3)
Mode	**ENUM**	Reset (0)	Oper	Oper	Senses	Reset (0) Run (1) Hold (2)
Output	**REAL**	0.0	Oper	Block	High Limit Low Limit	999,999 -99,999
Process_Val	**REAL**	0.0	Oper	Oper	High Limit Low Limit	999,999 -99,999
Ramp_Act	**BOOL**	No (0)	Oper	Block	Senses	No (0) Yes (1)
Ramp_End	**BOOL**	False (0)	Oper	Block	Senses	False (0) True (1)
Rate	**REAL**	0.0	Oper	Oper	High Limit Low Limit	100,000 0
Rate_Units	**ENUM**	/Second (0)	Oper	Oper	Enumerate d Values	/Second (0) /Minute (1) /Hour (2) /Day (3)
Reset_Output	**REAL**	0.0	Oper	Oper	High Limit Low Limit	999,999 -99,999
Setpoint	**REAL**	0.0	Oper	Oper	High Limit Low Limit	999,999 -99,999
Time_Remain	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms

Table 15-2 Ramp_Rate Parameter Attributes

PROG8RATE FUNCTION BLOCK

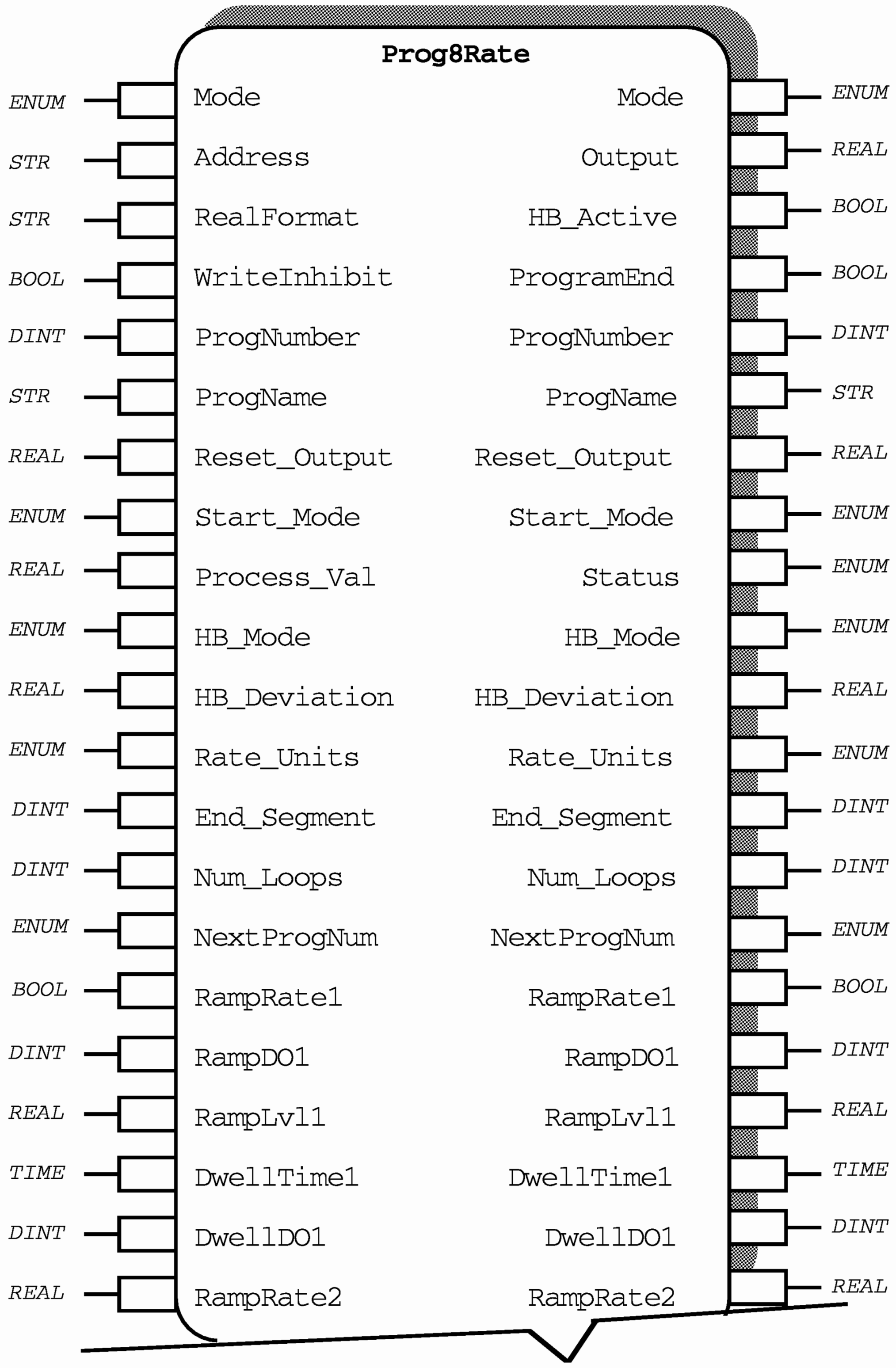


Figure 15-5 Prog8Rate Function Block Diagram

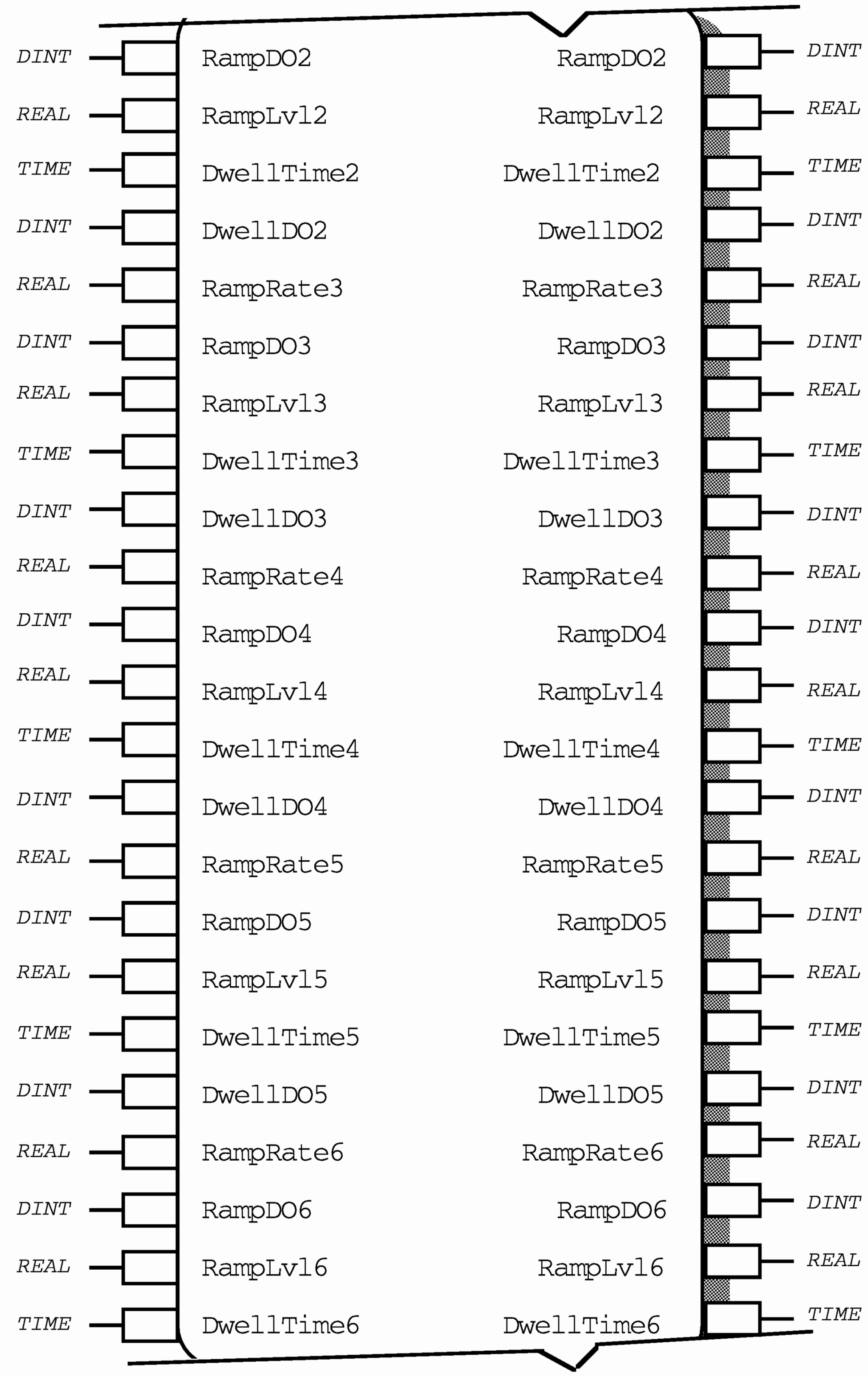

DINT RampDO2 RampDO2 DINT
REAL RampLvl2 RampLvl2 REAL
TIME DwellTime2 DwellTime2 TIME
DINT DwellDO2 DwellDO2 DINT
REAL RampRate3 RampRate3 REAL
DINT RampDO3 RampDO3 DINT
REAL RampLvl3 RampLvl3 REAL
TIME DwellTime3 DwellTime3 TIME
DINT DwellDO3 DwellDO3 DINT
REAL RampRate4 RampRate4 REAL
DINT RampDO4 RampDO4 DINT
REAL RampLvl4 RampLvl4 REAL
TIME DwellTime4 DwellTime4 TIME
DINT DwellDO4 DwellDO4 DINT
REAL RampRate5 RampRate5 REAL
DINT RampDO5 RampDO5 DINT
REAL RampLvl5 RampLvl5 REAL
TIME DwellTime5 DwellTime5 TIME
DINT DwellDO5 DwellDO5 DINT
REAL RampRate6 RampRate6 REAL
DINT RampDO6 RampDO6 DINT
REAL RampLvl6 RampLvl6 REAL
TIME DwellTime6 DwellTime6 TIME

Figure 15-5 Prog8Rate Function Block Diagram (continued)

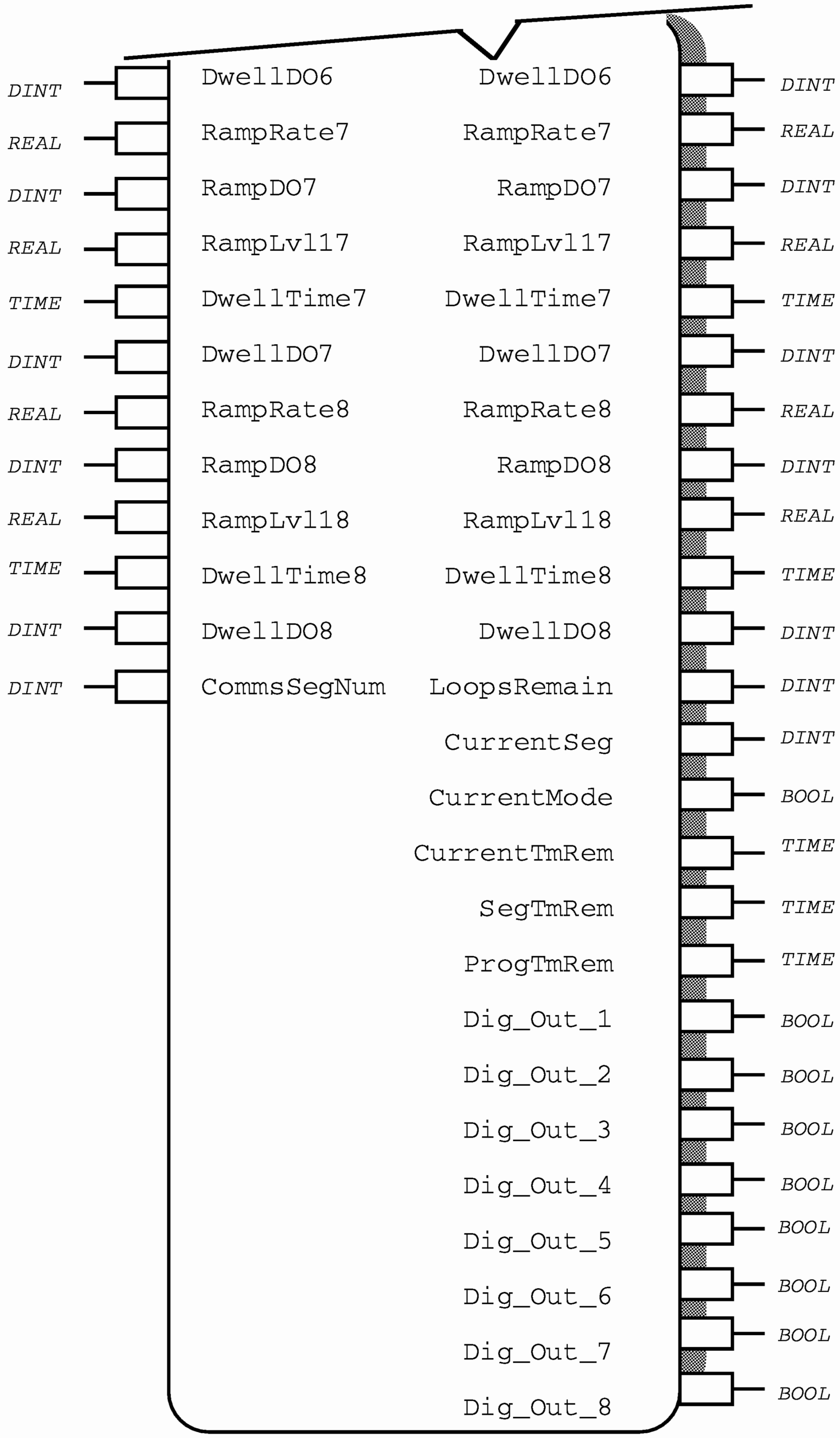


Figure 15-5 Prog8Rate Function Block Diagram

Functional Description

The **Prog8Rate** function block produces an **Output** that can be used as a setpoint profile to drive a control loop. By profile we mean a pre-defined variation of the setpoint as a function of time. This is similar to facilities provided bythe Eurotherm 818/902 multiprogrammer instruments.

It is possible to store profiles (with certain other related settings) as named programs in the PC3000 file store system.

A profile consists of up to eight segments - the number of segments in use in any given profile is specified by the input **End_Segment.** Each segment would normally be defined as a ramp followed by a dwell period, as in segments 1, 2, 3 and 8 of Figure 15-3

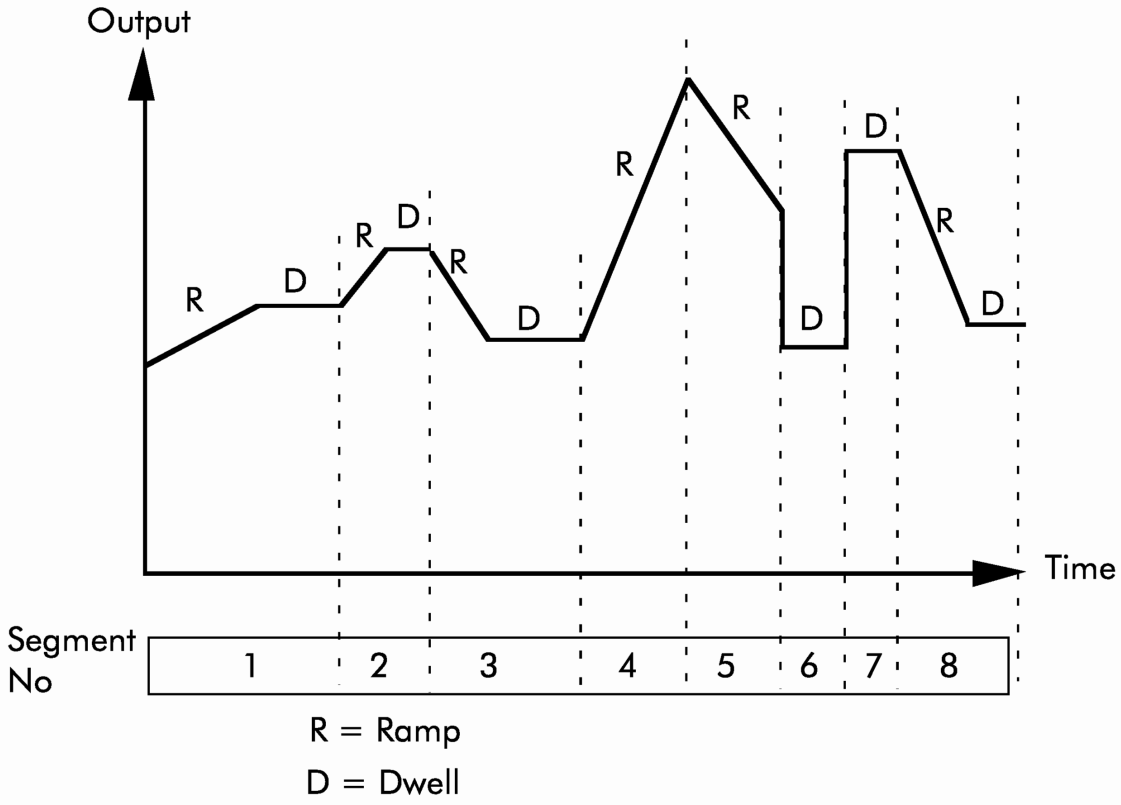


Figure 15-6 Setpoint Profile: Prog8Rate.Output as a function of Time

A segment can also be defined as only a ramp, as in segments 4 and 5 above, meaning that the next segment starts as soon as the ramp finishes. This allows peaks or changes of gradient to be created within a profile.

Alternatively, a segment may only be a dwell period, meaning any change in level of output is achieved as a step change. See segments 6 and 7 above.

A program will always start with segment 1 unless the parameter **Start_Mode** dictates otherwise. See later for more detail on this. The program will then proceed sequentially through each segment until the last configured segment is reached, or the eighth segment is reached. The current segment number and its mode (ramp or dwell) is reported.

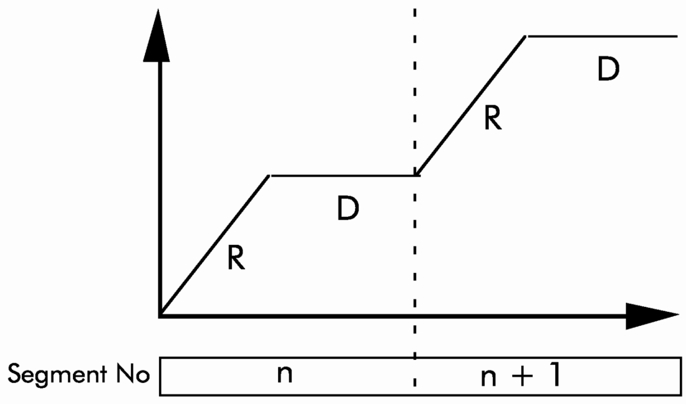
The starting value of **Output** can be preset by means of the input **Reset_Output.** Whenever the **Mode** input is set to Reset, the **Output** will be set to **Reset_Output.** A segment starts from the current value of **Output** (see Figure 15-4) and it may therefore be necessary to set mode to **Reset** and then back to **Run** in order that the first segment starts from a known position.

With the **Prog8Rate** function block, each ramp segment of a profile is specified by means of a ramp rate (**RampRate1** to **RampRate8**) and a target setpoint (**RampLvl1** to **RampLvl8**). The rate units e.g. per second, per hour etc. can also be defined.

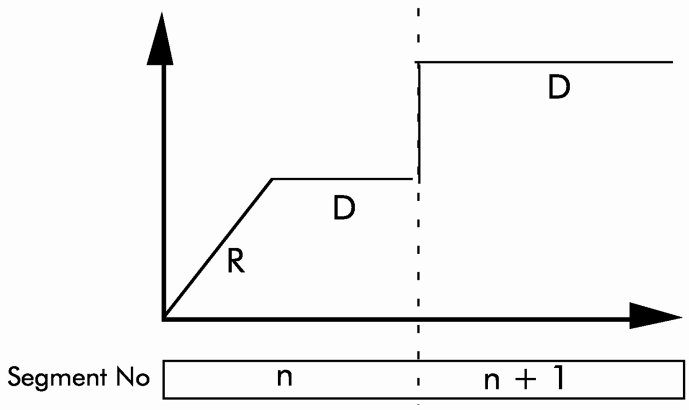
A profile may be repeated up to 999 times. A profile will always repeat by jumping from the configured last segment (or the eighth segment) back to segment 1. The current number of loops remaining (excluding the currently running loop) will always be reported. This means that when the segment indicated by **End_Segment** is completed, the Output will then be driven from its current position towards **RampLvl1** at the rate specified by **RampRate1.** In order to repeat a profile exactly therefore, the value of **RampLvl8** must be the same as the start value of **Output.**

A ramp rate of zero will be interpreted as a step or null ramp - the output will instantly be set to the target level. A dwell time of zero seconds will be interpreted as a null dwell. No time is required to execute a null ramp or a null dwell.

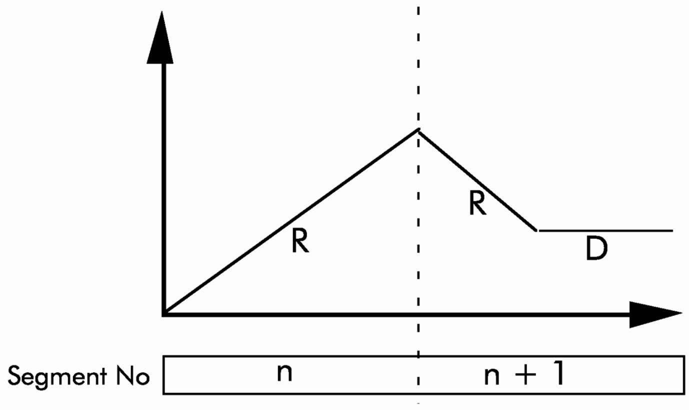
A segment will be completely skipped (or ignored) if a step (null ramp) and a null dwell are configured within that same segment. The outputs will not be affected in any way by a skipped segment.



Normal Ramp/Dwell sequence



A step can be achieved by setting the ramp rate in segment n + 1 to zero.



Two consecutive ramps can be achieved by setting the dwell time in segment n to zero.

There are no limits to the number of consecutive skipped segments, null ramps or null dwells. On completion of the previous non-null ramp or dwell, the programmer will immediately skip the null ramp(s) and null dwell(s) and proceed to execute the next non-null ramp or dwell, ignoring all null segments. If all eight segments are null, the program will immediately end irrespective of the number of loops remaining.

On completion of a profile (including any repeats), the **ProgramEnd** flag will become true. When a program ends, the states of all outputs will remain in the same state that they were in immediately prior to ending, until the programmer is reset.

A program can be cut short by setting the last segment that will executed by means of the **End_Segment** parameter. The default end segment will be segment 8. Any segments following the specified end segment will always be ignored - for example, if the **End_Segment** is 3, then segments 4-8 will be ignored and the program will end once segment 3 is completed. If repeats are configured the program will always repeat by returning to segment 1 from segment 3.

Concatenation of programs is possible using the **NextProgNum** parameter. On completion of the current program (i.e. when **ProgramEnd** becomes true), if the **NextProgNum** parameter is non-zero, the programmer will automatically be set to **HOLD,** the program corresponding to the program number given by the **NextProgNum** parameter will be loaded, and once loaded, the new program will be set to run according to the **Start_Mode** of the new program.

A list of program names can be set up using the **ProgName** and **ProgNumber** parameters.

To make the creation of standard "PID with programmer" type of control straightforward, holdback action is a built-in option of the **Prog8Rate** function block. This means that the Output of the **Prog8Rate** can be prevented from deviating by more than a preset amount from the current measurement of the physical variable being controlled. This latter value is fed back to the programmer through the input **Process_Val.** Three modes of holdback are supported, and holdback can be configured using the **HB_Mode** and **HB_Deviation** parameters.

A set of eight digital outputs are provided. The state of each of these outputs may be individually programmed for each ramp and dwell of each segment of a profile. Their states form part of the program. The digital outputs are only updated on starting a non-null (i.e. non zero) ramp or dwell, and are not affected by null ramp or dwell segments.

Outputs are provided to indicate the current program state, including time progress indicators for the current segment and the total program (including repeats), and an indication of the number of repeats remaining.

If any parameter changes, then all the times remaining except program time remaining will be immediately re-calculated and updated to the new time. Program time remaining is only re-calculated during reset, hold, track, or during a change of segment (including when going from a ramp to a dwell).

As well as being used to control the operation of the current program, the **Mode** parameter may be used to save and retrieve programs from the PC3000 file store. Programs are handled as text files. Each program is saved and loaded by its name: names are case sensitive, and consist of up to 8 characters, a full stop then a 3 character extension. This is very simular to the standard MSDOS filename system.

To avoid unneccessay CPU loading, the loading and saving of programs is spread over some 50 execution cycles, resulting in a typical delay of 5 seconds before a program is completely loaded (assuming a 100ms task rate). The Status output will report the progress of save or load operations.

A program consists of the following pieces of information:

HB_Mode

HB_Deviation

Rate_Units

Reset_Output

Num_Loops

End_Segment

NextProgNum

RampRate1 - RampRate8

RampDO1 - RampDO8

RampLvl1 - RampLvl8

DwellTime1 - DwellTime8

DwellDO1 - DwellDO8

When a program is retrieved from the PC3000 File Store, all the above parameters will be defined by the values contained in the stored file. It should be noted that a program saved by Prog8Rate cannot be loaded by Prog8Time and vice-versa.

Any alteration of profile parameters, either over the communications link (defined later) or by direct change from the user program will become effective immediately if the program is in **Run Mode**. Great care should therefore be exercised in ensuring that changes to program parameters are only made when no adverse affect will occur on the plant. During ramping, if the ramp rate or the target level are changed, the ramp will immediately be re-calculated from the current value of **Output** utilising the new rate and target level information. During dwelling, if the dwell time is changed, the dwell will immediately be re-calculated - the TOTAL dwell time will be that of the new requested dwell time, and if the new requested dwell time is less than the already elapsed dwell, the dwell will immediately end. If target level is changed whilst dwelling, the output level will immediately change to the new target level. However, the digital outputs will only be updated when entering into the given ramp or dwell - if the digital output configuration for the currently executing ramp or dwell is modified, the digital outputs will not be correspondingly updated until that segment is re-entered.

Built in to the **Prog8Rate** function block are a set of **Slave_Vars** that can be used to control the programmer and update programs. The current program can be edited via a communications link and stored to the PC3000 File Store. A program can be recalled from the File Store and set active or edited. The table 15-3 below indicates the parameters and their addresses.

Parameter Name	**Access**	**Data Type**	**Europanel Address**	**Bisync Address**	**JBus Address**
Mode	Read /Write	DINT	MD	00	00R1
Status	Read Only	DINT	SS	01	01R1
Program Number	Read/Write	DINT	PN	02	02R1
Edit Segment Number	Read/Write	DINT	SN	03	03R1
Edit Ramp rate	Read/Write	REAL	RR	04	04 (R2)
Edit Ramp Digital Output	Read/Write	BOOL 8	RD	05	05 (Bit Space)
Edit Ramp Level	Read/Write	REAL	RL	14	14 (R2)
Edit Dwell Time	Read/Write	TIME	DT	16	16 (R2)
Edit Dwell Digital Output	Read/Write	BOOL 8	DD	18	18 (Bit Space)
Reset Output	Read/Write	REAL	RO	27	27 (R2)
Start Mode	Read/Write	DINT	SM	29	29R1
Hold Back Mode	Read/Write	DINT	HM	30	30R1
Hold Back Deviation	Read/Write	REAL	HD	31	31 (R2)
End Segment Number	Read/Write	DINT	ES	33	33R1
Number of Loops	Read/Write	DINT	NL	34	34R1
Next Program Number	Read/Write	DINT	NP	35	35R1
Output	Read Only	REAL	OP	36	36 (R2)
Process Value	Read Only	REAL	PV	38	38 (R2)
Hold Back Active	Read Only	BOOL	HA	40	40 (Bit Space)
Current Active Segment	Read Only	DINT	CS	41	41R1
Current Active Mode	Read Only	DINT	CM	42	42R1
Current Time Remaining	Read Only	TIME	CT	43	43 (R2)
Segment Time Remaining	Read Only	TIME	ST	45	45 (R2)
Program Time Remaining	Read Only	TIME	PT	47	47 (R2)
Current Loops Remaining	Read Only	DINT	LR	49	49R1
Program Name	Read Only	STRING	NM	50	50R7
Current Digital Output	Read Only	STRING	DO	64	64R4

Table 15-3 Addresses of Internal Slave Parameters

The single 4 character **Address** input parameter is used to indicate which communications protocol is to be used, and to form the base address of all the internal slave variables. For example EPP1 would indicate that a Europanel was being used to communicate with the slaves (EP), and the variables to be used in OIFL would be prefixed with P1, e.g. P1MD, P1SS etc. Alternatively, EB01 would indicate that communications was being carried out using the Eurotherm Bisync Protocol (EB), and the UID is zero and the channel number is 1. In this instance floating point format can be changed using the **RealFormat** input parameter. See the PC3000 Communications Overview and related documents for detail on setting up PC3000 communications.

To access segment data using the slave variables, the edit segment number or **CommsSegNum** is used to point to the segment to be accessed, and the five segment data parameters are used to access **ramp rate, ramp digital** outputs, **ramp level, dwell time,** and **dwell** digital outputs. The read/write slave parameters can have their write access temporarily inhibited by using the **WriteInhibit** parameter.

Function Block Attributes

Type:	3F20
Class:	PROGRAMMER
Default Task:	Task_2
Short List:	Mode, Output, HB_Active, ProgramEnd
Memory Requirements:	5676 Bytes

Parameter Descriptions

Mode (MD)

This parameter controls the operation of the function block and is also used to transfer programs to and from the file store.

Reset(0): **Output** is set to the value of **Reset_Output** and digital outputs are all cleared to Off(0). Every execution cycle the current program is examined, and progress-reporting outputs are set to their start values - they report the first non-null ramp or dwell, current ramp or dwell time, current segment time, and total program run time. If no non-null segments exist, the **ProgramEnd** flag will become true and the program will be unable to run until a non-null segment is configured.

Run(1): **Output** is controlled according to the profile set up by **RampRate1** - **RampRate8, RampLvl1 - RampLvl8** and **DwellTime1 - DwellTime8.** This profile may be modified by the operation of hold back. If previously in **Reset Mode, Output** will start at the first non-null ramp or dwell. If the program contains all null-segments, the program will immediately end, irrespective of the number of loops configured.

If previously in Hold or Track mode, the function block will immediately resume executing the current ramp or dwell. The program will continue running until an end segment is reached and no next program is configured, when the program end flag will become true. When a program ends, all outputs will remain held in their state immediately prior to ending until the program is reset. New programs may be loaded whilst **Mode** is set to Run.

Hold(2): All outputs are frozen at their current values. The profile effectively has an indeterminate dwell period temporarily inserted at its current position. New programs may be loaded whilst **Mode** is set to Hold. A program can be placed in **Hold Mode** at any time. If previously in **Reset Mode**, the program will commence executing the current program as if the mode was set to Run, but will immediately be placed in **Hold Mode.**

Track(3): The action of the program on **Output** is interrupted, and **Output** is driven directly by **Process_Val.** When the **Mode** is set back to **Run, Output** moves from its current value towards the ramp level of the current segment at ramp rate of the current segment. When it reaches the appropriate ramp level, the program resumes.

If the program is in the ramp part of a segment when **Mode** is changed to Track, **CurrentMode** remains as Ramp, and the three time remaining outputs are continuously updated to give the time that would be required to ramp from the current value of **Output** to the current segment ramp level at the ramp rate of the current segment. The digital outputs display the states dictated by the **RampDO** value for the current segment.

If the program is in the dwell part of a segment when **Mode** is changed to Track, **CurrentMode** returns to Ramp, and the three time remaining outputs are continuously updated to give the time that would be required to ramp from the current value of **Output** to the current segment ramp level at the ramp rate of the current segment. The digital outputs display the states dictated by the **DwellDO** value for the current segment.

A program can be placed in Track Mode at any time. If previously in **Reset Mode**, the program will commence executing the current program as if the mode was set to Run, but will immediately be placed in **Track Mode**.

Skip Seg(4): While in **Run Mode**, causes the remains of the currently running segment to be skipped and the beginning of the next segment to start executing.

Skip segment can only be performed while in **Run mode**. If requested in any other mode, the skip segment request will be ignored and the previous mode resumed unaffected.

NxtUpSg(5): While in **Reset Mode,** will cause the program to immediately start running. However **Output** immediately tracks the **Process Value** input, and instead of starting at the first segment, will immediately start executing the first segment whose target level is GREATER than **Process Value. Mode** will immediately return to Run mode.

NxtDnSg(6):While in **Reset mode,** will cause the program to immediately start running. However the **Output** immediately tracks the **Process Value** input, and instead of starting at the first segment, will immediately start executing the first segment whose target level is LESS than **Process Value.** The **mode** will immediately return to run mode.

Load(7): Normally, this operation would only be undertaken in **Reset Mode.** The current settings of the program are overwritten by the values stored as a program in the PC3000 File Store under the name given in **ProgName.** After succesfully loading the program, **Mode** is automatically restored to Reset if that was the **Mode** when the load operation was requested.

During loading the **Mode** of the current program, is immediately changed to **Hold**. All outputs are held at their previous values. The **Status** pin reports the progress of the load If **Mode** was previously Reset, Hold or Track, then following the load, **Mode** will return to Reset, Hold or Track respectively.

If **Mode** was Run at the start of the load operation, then following the load the mode will always be set according to the **Start_Mode** of the newly loaded program. However if the previous program had ended (i.e. **ProgramEnd** is Yes prior to Load being selected), the new program will be loaded, and on completion of the load the mode will return to **Run mode**, but the program end flag will remain true, and all outputs will remain unchanged from the values pertaining prior to the load being requested.

While loading, any further changes to **Mode** will be completely ignored - on completion of the load, the mode will be set according to the state of **Mode** prior to the load request If a file of the name given does not exist in the File Store, or if the file name does not conform to File Store conventions, or if no File Store has been created, an error will be shown on **Status** and **Mode** is automatically restored to **Reset** without any action by the user program.

Save(8): The current settings of the program are saved into the PC3000 File Store under the name given in **ProgName** and **Mode** is automatically restored to its current setting without any affect on the current program.

If the name given does not conform to File Store conventions, or if no File Store has been created, an error will be shown on Status. The Status pin reports the progress of the save.

While saving, any changes to **Mode** will be actioned immediately - the save will not be affected. The only exception is a load request which will be completely ignored. If a program ends during saving and a next program is configured, the requested next program will be ignored and the program will immediately end as if no next program was configured.

Address (ADR)

The first two characters of this parameter designate which communications protocol will be used to communicate with the built-in slave variables of the function block. The second two characters define the base address of all the slave variables. For example, EB01 would define the use of the Eurotherm Bisync protocol with all slaves having a UID of zero and a channel number of one.

RealFormat (RLF)

Selects the format to be used for real numbers passed over the communications link when using Eurotherm Bisync protocol. See description of the EIBisync Slave function block for detail.

For use with the EuroPanel 2 function block, this parameter must be set to any single character to ensure correct display of the digital outputs.

WriteInhibit (WI)

It is possible to protect the program parameters from being written to when being addressed over a communications link by setting this input to **Rd_Only.** Program parameters can still be read over the communications link.

Setting this parameter to **Rd_Wr** permits program parameters to be both written and read over a communications link.

ProgNumber (PN)

The **Prog8Rate** function block retains a cross reference between 32 program numbers and 32 associated program names. When a given **ProgNumber** is selected, the program name last associated with this program number is displayed on the **ProgName** input/output.

ProgName (PNM)

An input/output used to enter the name of the program associated with the current **ProgNumber.** The name entered should be a valid file name for the PC3000 file system so that programs can be loaded from or saved to the file store.

Reset_Output (ROP)

The value that will appear on the Output when the **Mode** is set to Reset. Whilst the **Mode** remains in Reset, any change to **Reset_Output** is mirrored on Output.

Start_Mode (STM)

When a program is loaded, if **Mode** was Run at the start of the load operation, then following the load the mode will be set according to the **Start_Mode** of the newly loaded program. Possible values for **Start_Mode** are:

Reset(0):	After loading the program, Mode will be set to Reset.
Run(1):	After loading the program, Mode will be set to Run.
Hold(2):	After loading the program, Mode will be set to Hold.
Track(3):	After loading the program, Mode will be set to Track.
SkipSeg(4):	After loading the program, Mode will be set to SkipSeg.
NxtUpSeg(5):	After loading the program, Mode will be set to NxtUpSeg.
NxtDnSeg(6):	After loading the program, Mode will be set to NxtDnSeg.

However if the previous program had ended (i.e. **ProgramEnd** is Yes prior to load being selected), the new program will be loaded, and on completion of the load **Mode** will return to Run, but the program end flag will remain true and all outputs will remain unchanged from the values pertaining prior to the load being requested.

Process_Val (PV)

Normally the **Process_Val** of the PID loop being controlled by the Programmer function block would be wired to this input. This gives information to the function block that enables decisions to be taken based on the real world performance. **Output** can be set to track this input, or the ramp may be temporarily stopped if the process under control deviates from the setpoint by more than a given amount (this is know as "holdback"). It depends on the application whether a connection is needed to this input or not.

HB_Mode (HM)

By changing this input, the way in which holdback operates can be selected. Holdback operates during a ramp - there is no automatic hold-back during a dwell.

Off(0):	No holdback is in operation.
Lower(1):	When **HB_Mode** is set to Lower, holdback will become active if Process_Val is less than or equal to Output minus the **HB_Deviation.** In other words, the **Output** will stop at its current position until **Process_Val** increases. The output **HB_Active** will also be true under these circumstances. This mode is usually used in cases where there is

danger of the setpoint (i.e. the Output of the programmer function block) getting ahead of the process on rising ramps.

Upper(2): When **HB_Mode i**s set to Upper, holdback will become active if **Process_Val i**s greater than or equal to Output plus the **HB_Deviation.** In other words, the **Output** will stop at its current position until **Process_Val** decreases. The output **HB_Active** will also be true under these circumstances. This mode is usually used in cases where there is danger of the setpoint (i.e. the Output of the programmer function block) getting ahead of the process on falling ramps.

Band(3): If the absolute difference between **Process_Val** and Output is greater than or equal to **HB_Deviation,** holdback will become active if **HB_Mode** is set to Band. In other words, the Output will stop at its current position until this condition is no longer true. The output **HB_Active** will also be true under these circumstances. This mode is used to protect the process from lagging too far behind the setpoint (i.e. the Output of the programmer function block) on both rising and falling ramps.

HB_Deviation (HD)

Used in conjunction with **Process_Val** and **Output** to determine whether holdback should become active, depending on the setting of **HB_Mode.** It defines limits relative to Output within which **Process_Val** is expected to remain.

Rate_Units (RU)

This input is used to interpret the settings of **RampRate1 to RampRate8** as being either in output units per second, per minute, per hour or per day. All segments must use the same **Rate_Units.**

End_Segment (ES)

The last segment to be considered as part of the profile. This allows a profile to be brought to an end before all non-null segments have been executed. A change made to this parameter whilst a program is running will become effective if the segment made to be the end segment has not yet completed executing.

Num_Loops (NL)

The number of times the current profile will be repeated before passing to the next program or flagging that the program has ended if NextProgNum is None.

NextProgNum (NPN)

This input allows a new program to be loaded from the file store once the the current program has completed.

None(0): No program will be loaded when the current program completes.

Prog_1(1): The program whose name is given as **ProgName** when **ProgNumber**=1 will be loaded from the PC3000 File Store when the current program completes. If there is no **ProgName** associated with **ProgNumber**=1 then the **ProgStatus** will show **LoadErr** and **Mode** will go to Reset.

Prog_2(2): The program whose name is given as **ProgName** when **ProgNumber**=2 will be loaded from the PC3000 File Store when the current program completes. If there is no ProgName associated with **ProgNumber**=2 then the ProgStatus will show **LoadErr** and **Mode** will go to Reset.

"

etc up to

Prog_32 (32)

RampRate1 (RR1)

The rate at which Output will move from its current value to the ramp level of segment 1 in **Output units** per **Rate_Units.**

RampDO1 (RD1)

This gives the possibility of setting each of the eight digital outputs of the Prog8Rate function block to a particular state during the ramp of segment 1. The value of this variable is interpreted as an eight bit pattern, the lowest bit setting **Dig_Out_1,** and the highest bit setting **Dig_Out_8.**

RampLvl1 (RL1)

The level of **Output** which, when achieved, will cause the dwell of segment 1 to commence.

DwellTime1 (DT1)

The length of time for which the **Output** will remain static at **RampLvl1** after the ramp part of segment 1 is completed.

DwellDO1 (DD1)

This gives the possibility of setting each of the eight digital outputs of the Prog8Rate function block to a particular state during the dwell of segment 1. The value of this variable is interpreted as an eight bit pattern, the lowest bit setting **Dig_Out_1,** and the highest bit setting **Dig_Out_8.**

RampRate2 (RR2)

The rate at which **Output** will move from its current value to the ramp level of segment 2 in **Output units** per **Rate_Units.**

RampDO2 (RD2)

This gives the possibility of setting each of the eight digital outputs of the Prog8Rate function block to a particular state during the ramp of segment 2. The value of this variable is interpreted as an eight bit pattern, the lowest bit setting **Dig_Out_1,** and the highest bit setting **Dig_Out_8.**

RampLvl2 (RL2)

The level of **Output** which, when achieved, will cause the dwell of segment 2 to commence.

DwellTime2 (DT2)

The length of time for which the **Output** will remain static at **RampLvl2** after the ramp part of segment 2 is completed.

DwellDO2 (DD2)

This gives the possibility of setting each of the eight digital outputs of the Prog8Rate function block to a particular state during the dwell of segment 2. The value of this variable is interpreted as an eight bit pattern, the lowest bit setting **Dig_Out_1,** and the highest bit setting **Dig_Out_8.**

"

etc up to

DwellDO8 (DD8)

CommsSegNum

Defines which Ramp, Level Dwell parameter set the communications mnemonics are pointing to.

See Edit Segment Number in Table 15-3.

Output (OP)

The current value of the profile being produced by the program now running. This Output would normally be wired to the **Setpoint** input of a PID loop or used in some other way as a setpoint for control action.

HB_Active (HA)

If **HB_Active** is true this indicates that the value currently being fed to the **Process_Val** input deviates from the current value of **Output** by more than the amount defined by **HB_Mode** and **HB_Deviation.**

ProgramEnd (PE)

This output indicates when the current program has ended, or where multiple programs are chained together using **NextProgNum,** when the last program in a chain has ended.

Status (SS)

The success or otherwise of the last file operation to or from the PC3000 File Store is indicated by this output, or, during a file operation, the operation in progress is indicated.

Ok(0): The last file store operation was succesful, either save or load.

Saving(1): A save operation is in progress to the file store .

Loading(2):A load operation is in progress from the file store.

SaveErr(3): The last save to file store operation failed. This could be for a variety of reasons, such as no file store exists, the file store is full or the file name specified in the **ProgName** parameter is invalid.

LoadErr(4): The last load from file store operation failed. This could be for a variety of reasons, such as no file store exists or the file name specified in the **ProgName** parameter is invalid or is not a recipe file. This error will also be reported if a program saved by a Prog8Time function block is loaded.

LoopsRemain (LR)

The number of loops (repeats) of the current program that have yet to be run once the current loop is completed.

CurrentSeg (CS)

The number of the segment of the current program currently being executed. This could either be the ramp or dwell part of the segment.

CurrentMode (CM)

This indicates whether the Ramp part or the Dwell part of the current segment is being executed. Normally the ramp part of a segent is followed by the dwell, and then the next segment begins to execute.

However, if **Mode** is set to Track when the dwell has already commenced, the **CurrentMode** will revert to Ramp. When **Mode** is set to Run again, the **Output** will ramp to the current segment **RampLvl** value at the current segment RampRate and then **CurrentMode** will again indicate Dwell. This second dwell period will last for the full **DwellTime** of the current segment.

CurrentTmRem (CTR)

The time that will be required to complete the operation indicated by **CurrentMode** (i.e. a ramp or dwell) assuming their are no interruptions caused by hold back becoming active, or due to **Mode** being other than Run.

The calculations are based on the current value of **Output** and the profile parameters for the current segment.

If hold back becomes active, or **Mode** is changed, **CurrentTmRem** indicates the time required to complete the operation indicated by CurrentMode if hold back were to become immediately inactive, or **Mode** were to be immediately changed to Run.

SegTmRem (STR)

The time that will be required to complete the current segment assuming their are no interruptions caused by hold back becoming active, or due to **Mode** being other than Run. The calculations are based on the current value of **Output** and the profile parameters for the current segment.

If hold back becomes active, or **Mode** is changed, **SegTmRem** indicates the time required to complete the current segment if hold back were to become immediately inactive, or **Mode** were to be immediately changed to Run.

ProgTmRem (PTR)

The time that will be required to complete the remaining number of loops of the current profile assuming their are no interruptions caused by hold back becoming active, or due to **Mode** being other than Run. The calculations are based on the current value of **Output** and the profile parameters for the current segment together with the **Num_Loops** parameter.

If hold back becomes active, or **Mode** is changed, **ProgTmRem** indicates the time required to complete the current segment if hold back were to become immediately inactive, or **Mode** were to be immediately changed to Run.

When **Mode** is set to Run and changes are made to program parameters, **ProgTmRem** will only be recalculated at the start of the next segment or at the changeover from a ramp to a dwell.

Dig_Out_1 (DO1)

A digital output which is driven by bit 0 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 0 of **RampDO4** is 1 and bit 0 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_1** will be switched on, and during the dwell in segment 4 **Dig_Out_1** will be switched off.

Dig_Out_2 (DO2)

A digital output which is driven by bit 1 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 1 of **RampDO4** is 1 and bit 1 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_2** will be switched on, and during the dwell in segment 4 **Dig_Out_2** will be switched off.

Dig_Out_3 (DO3)

A digital output which is driven by bit 2 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 2 of RampDO4 is 1 and bit 2 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_3** will be switched on, and during the dwell in segment 4 **Dig_Out_3** will be switched off.

Dig_Out_4 (DO4)

A digital output which is driven by bit 3 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 3 of **RampDO4** is 1 and bit 3 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_4** will be switched on, and during the dwell in segment 4 **Dig_Out_4** will be switched off.

Dig_Out_5 (DO5)

A digital output which is driven by bit 4 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 4 of **RampDO4** is 1 and bit 4 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_5** will be switched on, and during the dwell in segment 4 **Dig_Out_5** will be switched off.

Dig_Out_6 (DO6)

A digital output which is driven by bit 5 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 5 of **RampDO4** is 1 and bit 5 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_6** will be switched on, and during the dwell in segment 4 **Dig_Out_6** will be switched off.

Dig_Out_7 (DO7)

A digital output which is driven by bit 6 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 6 of RampDO4 is 1 and bit 6 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_7** will be switched on, and during the dwell in segment 4 **Dig_Out_7** will be switched off.

Dig_Out_8 (DO8)

A digital output which is driven by bit 7 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 7 of **RampDO4** is 1 and bit 7 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_8** will be switched on, and during the dwell in segment 4 **Dig_Out_8** will be switched off.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Mode	**ENUM**	Reset (0)	Oper	Oper	See parameter list	
Address	**STR**	'EPP1'	Super	Super	Max 4 characters	
RealFormat	**STR**	''	Super	Super	Single character	
WriteInhibit	**BOOL**	Rd_Wr(0)	Super	Super	Senses	Rd-Wr (0)
						Rd_Only (1)
ProgNumber	**DINT**	1	Oper	Oper	High Limit	32
					Low Limit	1
ProgName	**STR**	''	Oper	Oper	Max 12 characters	
Reset_Output	**REAL**	0	Super	Super	High Limit	+3.402823E+38
					Low Limit	-3.402823E+38
Start_Mode	**ENUM**	Run (1)	Oper	Oper	See parameter list	
Process_Val	**REAL**	0	Oper	Oper	High Limit	+3.402823E+38
					Low Limit	-3.402823E+38
HB_Mode	**ENUM**	Off (0)	Oper	Oper	See parameter list	
HB_Deviation	**REAL**	0	Oper	Oper	See parameter list	
Rate_Units	**ENUM**	/Second (0)	Super	Super	See parameter list	
End_Segment	**DINT**	8	Super	Super	High Limit	8
					Low Limit	1
Num_Loops	**DINT**	1	Super	Super	High Limit	999
					Low Limit	0
NextProgNum	**ENUM**	None(0)	Super	Super	See parameter list	
RampRate1 -to RampRate8	**REAL**	0	Super	Super	High Limit	+3.402823E+38
					Low Limit	-3.402823E+38
RampDO1 to RampDO8	**DINT**	0	Super	Super	High Limit	256
					Low Limit	0
RampLv11 to RampLv18	**REAL**	0	Super	Super	High Limit	+3.402823E+38
					Low Limit	-3.402823E+38
DwellTime1 to DwellTime8	**TIME**	0ms	Super	Super	High Limit	23d23h59m59s999ms
					Low Limit	0ms
DwellD01-to DwellD08	**DINT**	0	Super	Super	High Limit	256
					Low Limit	0
CommsSegNum	**DINT**	1	Super	Super	High Limit	8
					Low Limit	1

Table 15-4 Prog8Rate Parameter Attributes (continued)

Name	**Type**	**Cold Start**	**Read Access**	**Write Access**	**Type Specific Information**	
Output	**REAL**	0	Oper	Block	High Limit Low Limit	+3.402823E+38 -3.402823E+38

HB_Active	**BOOL**	No(0)	Oper	Block	Senses	No(0)
						Yes(1)

ProgramEnd	**BOOL**	Yes(1)	Oper	Block	Senses	No(0)
						Yes(1)

Status	**ENUM**	Ok(0)	Oper	Block	See parameter list	
LoopsRemain	**DINT**	0	Oper	Block	High Limit Low Limit	999 0
CurrentSeg	**DINT**	8	Oper	Block	High Limit Low Limit	8 1

CurrentMode	**BOOL**	Dwell(1)	Oper	Block	Senses	Ramp(0)
						Dwell(1)

CurrentTmRem	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0sms
SegTmRem	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms
ProgTmRem	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms

Dig_Out_1 -to Dig_Out_8	**BOOL**	Off(0)	Oper	Block	Senses	Off(0)
						On(1)

Table 15-4 Prog8Rate Parameter Attributes

PROG8TIME FUNCTION BLOCK

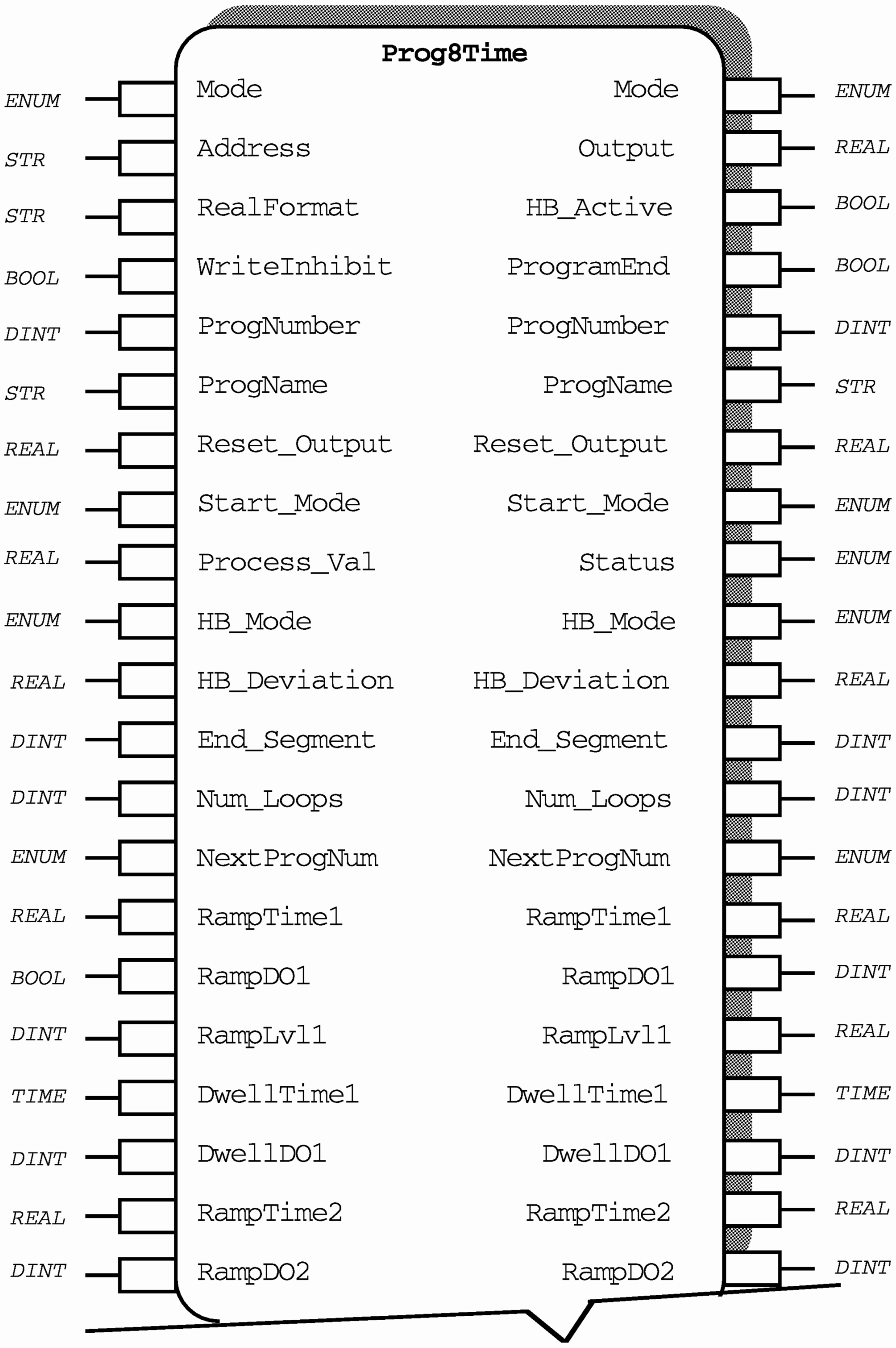
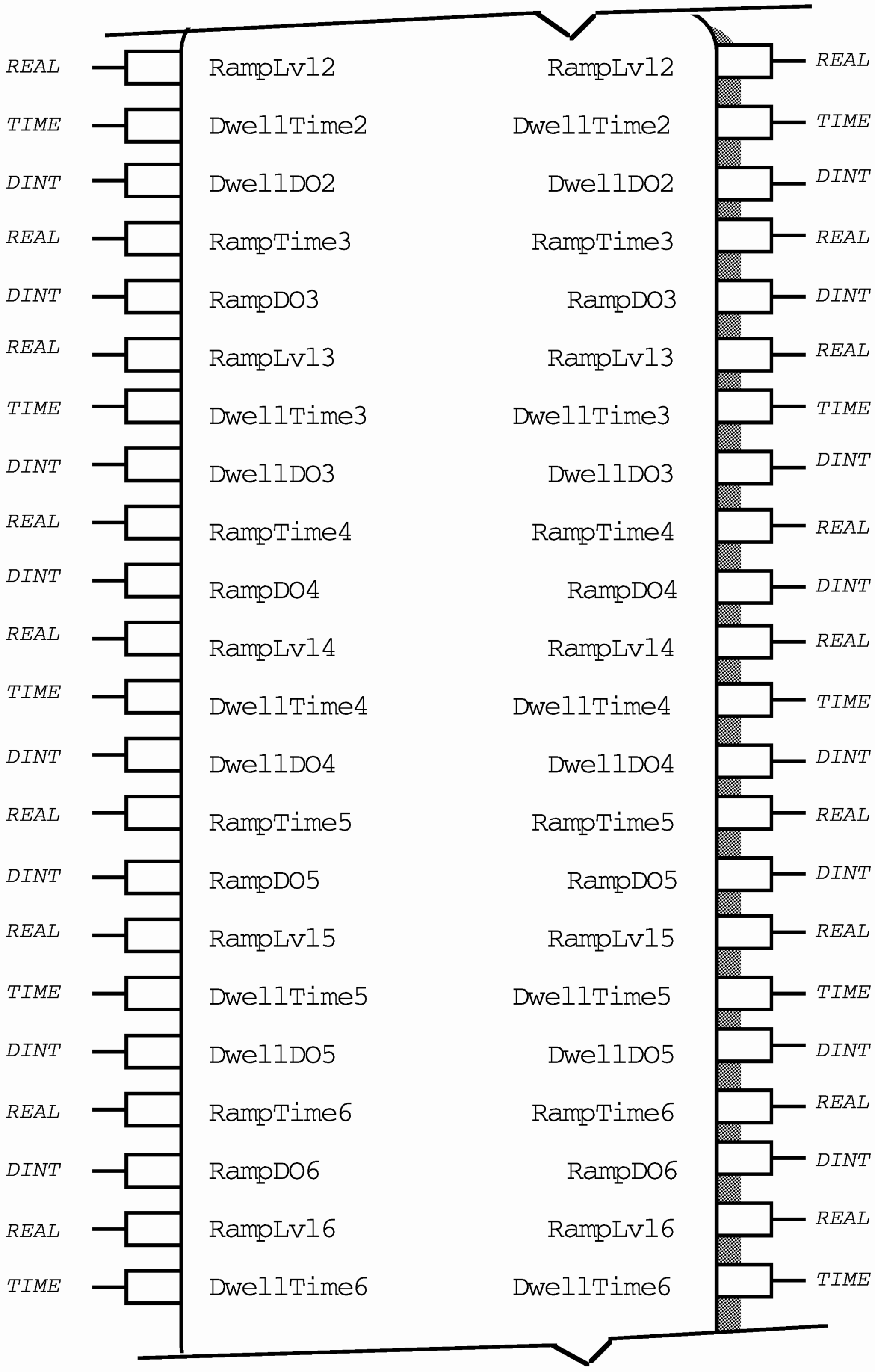

Prog8Time
ENUM Mode Mode ENUM
STR Address Output REAL
STR RealFormat HB_Active BOOL
BOOL WriteInhibit ProgramEnd BOOL
DINT ProgNumber ProgNumber DINT
STR ProgName ProgName STR
REAL Reset_Output Reset_Output REAL
ENUM Start_Mode Start_Mode ENUM
REAL Process_Val Status ENUM
ENUM HB_Mode HB_Mode ENUM
REAL HB_Deviation HB_Deviation REAL
DINT End_Segment End_Segment DINT
DINT Num_Loops Num_Loops DINT
ENUM NextProgNum NextProgNum ENUM
REAL RampTime1 RampTime1 REAL
BOOL RampDO1 RampDO1 DINT
DINT RampLvl1 RampLvl1 REAL
TIME DwellTime1 DwellTime1 TIME
DINT DwellDO1 DwellDO1 DINT
REAL RampTime2 RampTime2 REAL
DINT RampDO2 RampDO2 DINT

Figure 15-8 Prog8Time Function Block Diagram



REAL RampLvl2 RampLvl2 REAL
TIME DwellTime2 DwellTime2 TIME
DINT DwellDO2 DwellDO2 DINT
REAL RampTime3 RampTime3 REAL
DINT RampDO3 RampDO3 DINT
REAL RampLvl3 RampLvl3 REAL
TIME DwellTime3 DwellTime3 TIME
DINT DwellDO3 DwellDO3 DINT
REAL RampTime4 RampTime4 REAL
DINT RampDO4 RampDO4 DINT
REAL RampLvl4 RampLvl4 REAL
TIME DwellTime4 DwellTime4 TIME
DINT DwellDO4 DwellDO4 DINT
REAL RampTime5 RampTime5 REAL
DINT RampDO5 RampDO5 DINT
REAL RampLvl5 RampLvl5 REAL
TIME DwellTime5 DwellTime5 TIME
DINT DwellDO5 DwellDO5 DINT
REAL RampTime6 RampTime6 REAL
DINT RampDO6 RampDO6 DINT
REAL RampLvl6 RampLvl6 REAL
TIME DwellTime6 DwellTime6 TIME

Figure 15-8 Prog8Time Function Block Diagram (continued)

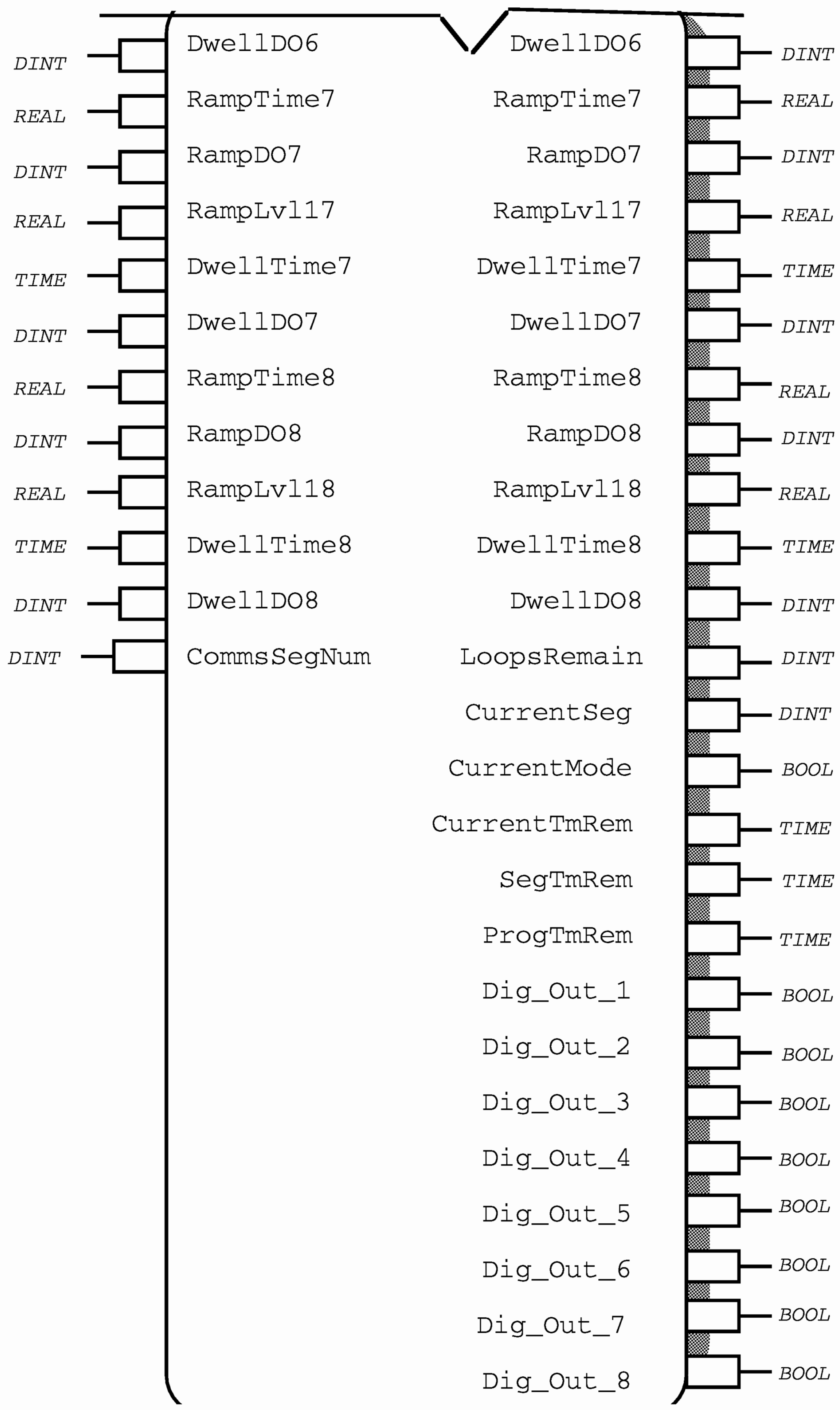


Figure 15-8 Prog8Time Function Block Diagram

Functional Description

The **Prog8Time** function block produces an **Output** that can be used as a setpoint profile to drive a control loop. By profile we mean a pre-defined variation of the setpoint as a function of time. This is similar to facilities provided by the Eurotherm 818/902 multiprogrammer instruments.

It is possible to store profiles (with certain other related settings) as named programs in the PC3000 file store system.

A profile consists of up to eight segments - the number of segments in use in any given profile is specified by the input **End_Segment.** Each segment would normally be defined as a ramp followed by a dwell period, as in segments 1, 2, 3 and 8 of diagram 15-9 below.

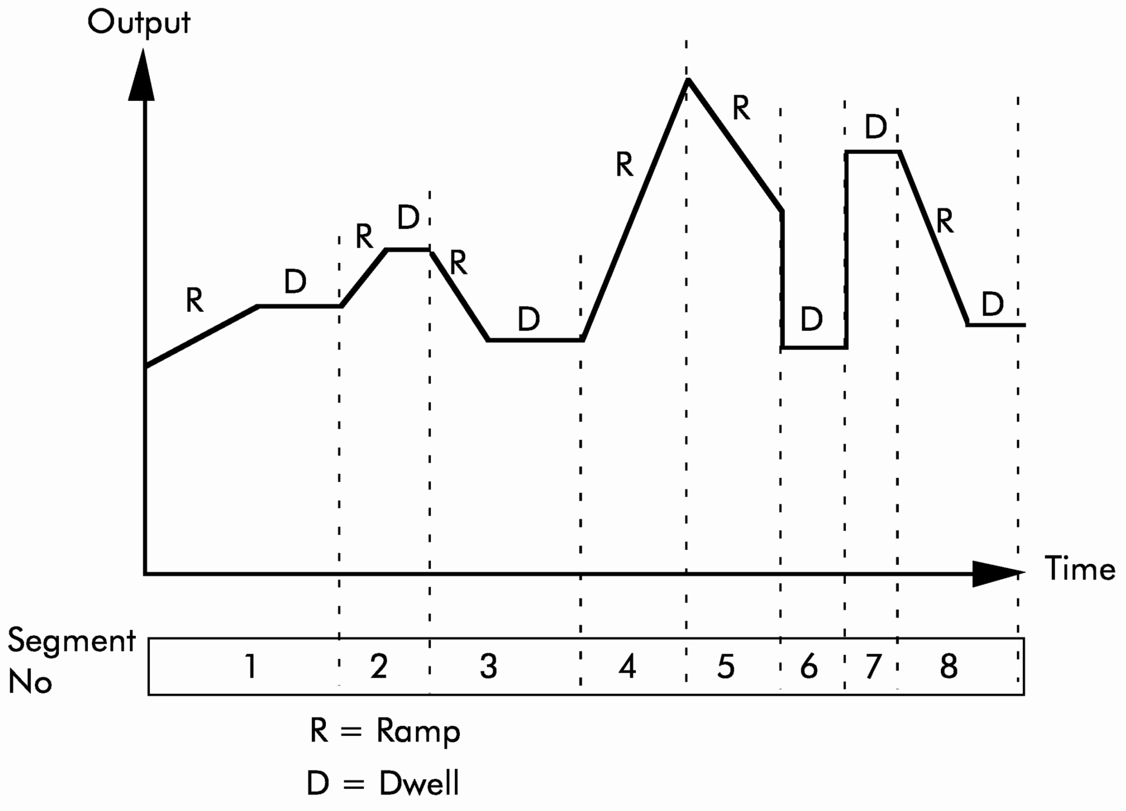


Figure 15-9 Setpoint Profile: Prog8Rate.Output as a function of Time

A segment can also be defined as only a ramp, as in segments 4 and 5 above, meaning that the next segment starts as soon as the ramp finishes. This allows peaks or changes of gradient to be created within a profile.

Alternatively, a segment may only be a dwell period, meaning any change in level of output is achieved as a step change. See segments 6 and 7 above.

A program will always start with segment 1 unless the parameter **Start_Mode** dictates otherwise. See later for more detail on this. The program will then proceed sequentially through each segment until the last configured segment is reached, or the eighth segment is reached. The current segment number and its mode (ramp or dwell) is reported.

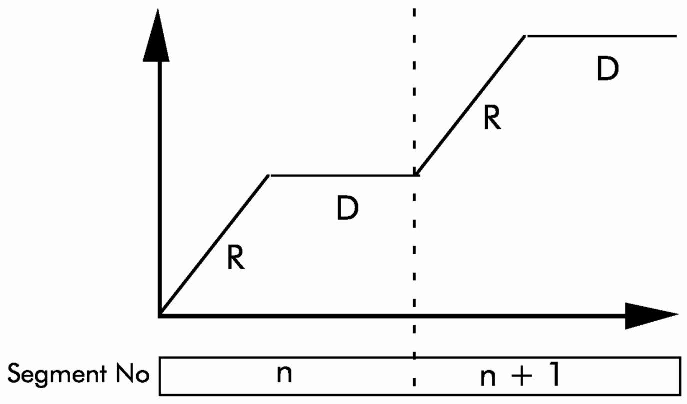
The starting value of **Output** can be preset by means of the input **Reset_Output.** Whenever the Mode input is set to Reset, the **Output** will be set to **Reset_Output.** A segment starts from the current value of **Output** (see figure 15-11) and it may therefore be necessary to set mode to Reset and then back to Run in order that the first segment starts from a known position.

With the **Prog8Time** function block, each ramp segment of a profile is specified by means of the time to complete the ramp (**RampTime1** to **RampTime8**) and the target setpoint (**RampLvl1** to **RampLvl8**).

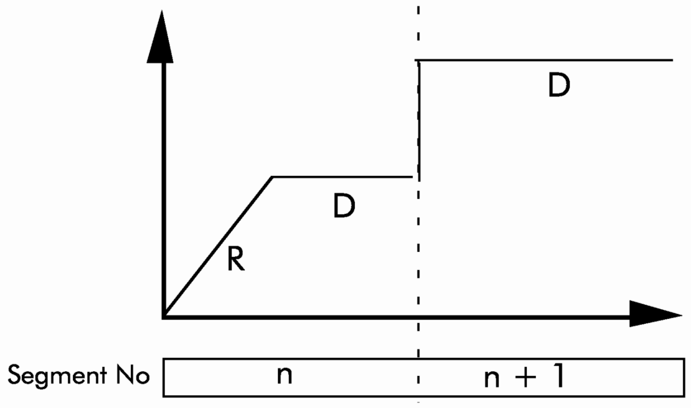
A profile may be repeated up to 999 times. A profile will always repeat by jumping from the configured last segment (or the eighth segment) back to segment 1. The current number of loops remaining (excluding the currently running loop) will always be reported. This means that when the segment indicated by **End_Segment** is completed, the **Output** will then be driven from its current position towards RampLvl1 in the time specified by **RampTime1.** In order to repeat a profile exactly therefore, the value of **RampLvl8** must be the same as the start value of **Output.**

A ramp time of zero will be interpreted as a step or null ramp - the output will instantly be set to the target level. A dwell time of zero seconds will be interpreted as a null dwell. No time is required to execute a null ramp or a null dwell.

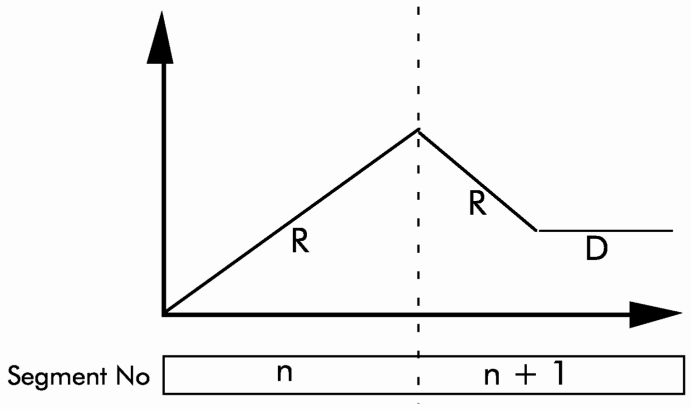
A segment will be completely skipped (or ignored) if a step (null ramp) and a null dwell are configured within that same segment. The outputs will not be affected in any way by a skipped segment.



Normal Ramp/Dwell sequence



A step can be achieved by setting the ramp time in segment n + 1 to zero.



Two consecutive ramps can be achieved by setting the dwell time in segment n to zero.

There are no limits to the number of consecutive skipped segments, null ramps or null dwells. On completion of the previous non-null ramp or dwell, the programmer will immediately skip the null ramp(s) and null dwell(s) and proceed to execute the next non-null ramp or dwell, ignoring all null segments. If all eight segments are null, the program will immediately end irrespective of the number of loops remaining.

On completion of a profile (including any repeats), the **ProgramEnd** flag will become true. When a program ends, the states of all outputs will remain in the same state that they were in immediately prior to ending, until the programmer is reset.

A program can be cut short by setting the last segment that will executed by means of the **End_Segment** parameter. The default end segment will be segment 8. Any segments following the specified end segment will always be ignored - for example, if the **End_Segment** is 3, then segments 4-8 will be ignored and the program will end once segment 3 is completed. If repeats are configured the program will always repeat by returning to segment 1 from segment 3.

Concatenation of programs is possible using the **NextProgNum** parameter. On completion of the current program (i.e. when **ProgramEnd** becomes true), if the **NextProgNum** parameter is non-zero, the programmer will automatically be set to HOLD, the program corresponding to the program number given by the **NextProgNum** parameter will be loaded, and once loaded, the new program will be set to run according to the **Start_Mode** of the new program.

A list of program names can be set up using the **ProgName** and **ProgNumber** parameters.

To make the creation of standard "PID with programmer" type of control straightforward, holdback action is a built-in option of the **Prog8Rate** function block. This means that the **Output** of the **Prog8Rate** can be prevented from deviating by more than a preset amount from the current measurement of the physical variable being controlled. This latter value is fed back to the programmer through the input **Process_Val..** Three modes of holdback are supported, and holdback can be configured using the **HB_Mode** and **HB_Deviation** parameters.

A set of eight digital outputs are provided. The state of each of these outputs may be individually programmed for each ramp and dwell of each segment of a profile. Their states form part of the program. The digital outputs are only updated on Starting a non-null (i.e. non zero) ramp or dwell, and are not affected by null ramp or dwell segments.

Outputs are provided to indicate the current program state, including time progress indicators for the current segment and the total program (including repeats), and an indication of the number of repeats remaining.

If any parameter changes, then all the times remaining except program time remaining will be immediately re-calculated and updated to the new time.Program time remaining is only re-calculated during reset, hold, track, or during a change of segment (including when going from a ramp to a dwell).

As well as being used to control the operation of the current program, the **Mode** parameter may be used to save and retrieve programs from the PC3000 file store. Programs are handled as text files. Each program is saved and loaded by its name: names are case sensitive, and consist of up to 8 characters, a full stop then a 3 character extension. This is very similar to the standard MSDOS filename system. To avoid unneccessay CPU loading, the loading and saving of programs is spread over some 50 execution cycles, resulting in a typical delay of 5 seconds before a program is completely loaded (assuming a 100ms task rate). The Status output will report the progress of save or load operations.

A program consists of the following pieces of information:

HB_Mode

HB_Deviation

Rate_Units

Num_Loops

End_Segment

NextProgNum

RampTime1 - RampTime8

RampDO1 - RampDO8

RampLvl1 - RampLvl8

DwellTime1 - DwellTime8

DwellDO1 - DwellDO8

When a program is retrieved from the PC3000 File Store, all the above parameters will be defined by the values contained in the stored file. It should be noted that a program saved by Prog8Rate cannot be loaded by Prog8Time and vice-versa.

Any alteration of profile parameters, either over the communications link (defined later) or by direct change from the user program will become effective immediately if the program is in **Run Mode.** Great care should therefore be exercised in ensuring that changes to program parameters are only made when no adverse affect will occur on the plant. During ramping, if the ramp time or the target level are changed, the ramp will immediately be re-calculated from the current value of Output utilising the new ramp time and target level information. During dwelling, if the dwell time is changed, the dwell will be re-calculated immediately - the TOTAL dwell time will be that of the new requested dwell time, and if the new requested dwell time is less than the already elapsed dwell, the dwell will immediately end. If target level is changed whilst dwelling, the output level will immediately change to the new target level. However, the digital outputs will only be updated when entering into the given ramp or dwell - if the digital output configuration for the currently executing ramp or dwell is modified, the digital outputs will not be correspondingly updated until that segment is re-entered.

Built in to the **Prog8Time** function block are a set of **Slave_Vars** that can be used to control the programmer and update programs. The current program can be edited via a communications link and stored to the PC3000 File Store. A program can be recalled from the File Store and set active or edited. The table below indicates the parameters and their addresses.

Parameter Name	Access	Data Type	Europane I Address	Bisync Address	JBUS Address
Mode	Read /Write	DINT	MD	00	00R1
Status	Read Only	DINT	SS	01	01R1
Program Number	Read/Write	DINT	PN	02	02R1
Edit Segment Number	Read/Write	DINT	SN	03	03R1
Edit Ramp Time	Read/Write	TIME	RT	04	04 (R2)
Edit Ramp Digital Output	Read/Write	BOOL 8	RD	05	05 (Bit Space)
Edit Ramp Level	Read/Write	REAL	RL	14	14 (R2)
Edit Dwell Time	Read/Write	TIME	DT	16	16 (R2)
Edit Dwell Digital Output	Read/Write	BOOL 8	DD	18	18 (Bit Space)
Reset Output	Read/Write	REAL	RO	27	27 (R2)
Start Mode	Read/Write	DINT	SM	29	29R1
Hold Back Mode	Read/Write	DINT	HM	30	30R1
Hold Back Deviation	Read/Write	REAL	HD	31	31 (R2)
End Segment Number	Read/Write	DINT	ES	33	33R1
Number of Loops	Read/Write	DINT	NL	34	34R1
Next Program Number	Read/Write	DINT	NP	35	35R1
Output	Read Only	REAL	OP	36	36 (R2)
Process Value	Read Only	REAL	PV	38	38 (R2)
Hold Back Active	Read Only	BOOL	HA	40	40 (Bit Space)
Current Active Segment	Read Only	DINT	CS	41	41R1
Current Active Mode	Read Only	DINT	CM	42	42R1
Current Time Remaining	Read Only	TIME	CT	43	43 (R2)
Segment Time Remaining	Read Only	TIME	ST	45	45 (R2)
Program Time Remaining	Read Only	TIME	PT	47	47 (R2)
Current Loops Remaining	Read Only	DINT	LR	49	49R1
Program Name	Read Only	STRING	NM	50	50R7
Current Digital Output	Read Only	STRING	DO	64	64R4

Table 15-5 Addresses of Internal Slave Parameters

The single 4 character **Address** input parameter is used to indicate which communications protocol is to be used, and to form the base address of all the internal slave variables. For example EPP1 would indicate that a Europanel was being used to communicate with the slaves (EP), and the variables to be used in OIFL would be prefixed with P1, e.g. P1MD, P1SS etc. Alternatively, EB01 would indicate that communications was being carried out using the Eurotherm Bisync Protocol (EB), and the UID is zero and the channel number is 1. In this instance floating point format can be changed using the RealFormat input parameter. See the PC3000 Communications Overview and related documents for detail on setting up PC3000 communications.

To access segment data using the slave variables, the edit segment number or **CommsSegNum** is used to point to the segment to be accessed, and the five segment data parameters are used to access ramp rate, ramp digital outputs, ramp level, dwell time, and dwell digital outputs. The read/write slave parameters can have their write access temporarily inhibited by using the **WriteInhibit** parameter.

Function Block Attributes

Type:	3F22
Class:	PROGRAMMER
Default Task:	Task_2
Short List:	Mode, Output, HB_Active, ProgramEnd
Memory Requirements:	5640 Bytes

Parameter Descriptions

Mode (MD)

This parameter controls the operation of the function block and is also used to transfer programs to and from the file store.

Reset(0): Output is set to the value of **Reset_Output** and digital outputs are all cleared of Off(0). Every execution cycle the current program is examined, and progress-reporting outputs are set to their start values - they report the first non-null ramp or dwell, current ramp or dwell time, current segment time, and total program run time. If no non-null segments exist, the ProgramEnd flag will become true and the program will be unable to run until a non-null segment is configured.

Run(1): Output is controlled according to the profile set up by **RampTime1 - RampTime8, RampLvl1 - RampLvl8** and **DwellTime1 -**

DwellTime8. This profile may be modified by the operation of hold back.

If previously in **Reset Mode,** Output will start at the first non-null ramp or dwell. If the program contains all null-segments, the program will immediately end, irrespective of the number of loops configured.

If previously in **Hold** or **Track mode,** the function block will immediately resume executing the current ramp or dwell.

The program will continue running until an end segment is reached and no next program is configured, when the program end flag will become true. When a program ends, all outputs will remain held in their state immediately prior to ending until the program is reset.

New programs may be loaded whilst **Mode** is set to **Run.**

Hold(2): All outputs are frozen at their current values. The profile effectively has an indeterminate dwell period temporarily inserted at its current position.

New programs may be loaded whilst **Mode** is set to **Hold**.

A program can be placed in **Hold Mode** at any time. If previously in **Reset Mode,** the program will commence executing the current program as if the mode was set to **Run,** but will immediately be placed in **Hold Mode**.

Track(3): The action of the program on **Output** is interrupted, and **Output** is driven directly by **Process_Val.** When the **Mode** is set back to **Run**, Output moves from its current value towards the ramp level of the current segment in the ramp time of the current segment. When it reaches the appropriate ramp level, the program resumes.

If the program is in the ramp part of a segment when **Mode** is changed to Track, CurrentMode remains as Ramp, and the three time remaining outputs are continuously updated to give the time that would be required to ramp from the current value of **Output** to the current segment ramp level in the ramp time of the current segment. The digital outputs display the states dictated by the **RampDO** value for the current segment.

If the program is in the dwell part of a segment when **Mode** is changed to Track, **CurrentMode** returns to Ramp, and the three time remaining outputs are continuously updated to give the time that would be required to ramp from the current value of Output to the current segment ramp level in the ramp time of the current segment. The digital outputs display the states dictated by the **DwellDO** value for the current segment.

A program can be placed in Track Mode at any time. If previously in **Reset Mode**, the program will commence executing the current program as if the mode was set to Run, but will immediately be placed in **Track Mode**.

Skip Seg(4):While in **Run Mode**, causes the remains of the currently running segment to be skipped and the beginning of the next segment to start executing.

Skip segment can only be performed while in **Run mode**. If requested in any other mode, the skip segment request will be ignored and the previous mode resumed unaffected.

NxtUpSg(5): While in **Reset mode,** will cause the program to immediately start running. However **Output** immediately tracks the **Process Value** input, and instead of starting at the first segment, will immediately start executing the first segment whose target level is GREATER than **Process Value Mode** will immediately return to Run mode.

NxtDnSg(6):While in **Reset mode,** will cause the program to immediately start running. However the **Output** immediately tracks the **Process Value** input, and instead of starting at the first segment, will immediately start executing the first segment whose target level is LESS than **Process Value.** The mode will immediately return to **Run mode.**

Load(7): Normally, this operation would only be undertaken in **Reset Mode.** The current settings of the program are overwritten by the values stored as a program in the PC3000 File Store under the name given in **ProgName**. After succesfully loading the program, **Mode** is automatically restored to **Reset** if that was the **Mode** when the load operation was requested.

During loading the **Mode** of the current program, is immediately changed to **Hold**. All outputs are held at their previous values. The Status pin reports the progress of the load.

If **Mode** was previously **Reset, Hold** or **Track,** then following the load, **Mode** will return to **Reset, Hold** or **Track** respectively.

If **Mode** was **Run** at the start of the load operation, then following the load the mode will always be set according to the **Start_Mode** of the newly loaded program. However if the previous program had ended (i.e. **ProgramEnd** is Yes prior to **Load** being selected), the new program will be loaded, and on completion of the load the mode will return to **Run Mode**, but the program end flag will remain true, and all outputs will remain unchanged from the values pertaining prior to the load being requested.

While loading, any further changes to **Mode** will be completely ignored - on completion of the load, the mode will be set according to the state of **Mode** prior to the load request.

If a file of the name given does not exist in the File Store, or if the file name does not conform to File Store conventions, or if no File Store has been created, an error will be shown on **Status** and **Mode** is automatically restored to Reset without any action by the user program.

Save(8): The current settings of the program are saved into the PC3000 File Store under the name given in **ProgName** and **Mode** is automatically restored to its current setting without any affect on the current program.

If the name given does not conform to File Store conventions, or if no File Store has been created, an error will be shown on Status. The Status pin reports the progress of the save.

While saving, any changes to **Mode** will be actioned immediately - the save will not be affected. The only exception is a load request which will be completely ignored. If a program ends during saving and a next program is configured, the requested next program will be ignored and the program will immediately end as if no next program was configured.

Address (ADR)

The first two characters of this parameter designate which communications protocol will be used to communicate with the built-in slave variables of the function block. The second two characters define the base address of all the slave variables. For example, EB01 would define the use of the Eurotherm Bisync protocol with all slaves having a UID of zero and a channel number of one.

RealFormat(RLF)

Selects the format to be used for real numbers passed over the communications link when using Eurotherm Bisync protocol. See description of the EIBisync Slave function block for detail.

For use with the EuroPanel 2 function block this parameter must be set to any single character to ensure correct display of the digital outputs.

WriteInhibit (WI)

It is possible to protect the program parameters from being written to when being addressed over a communications link by setting this input to **Rd_Only. Program** parameters can still be read over the communications link.

Setting this parameter to **Rd_Wr** permits program parameters to be both written and read over a communications link.

ProgNumber (PN)

The **Prog8Rate** function block retains a cross reference between 32 program numbers and 32 associated program names. When a given **ProgNumber** is selected, the program name last associated with this program number is displayed on the **ProgName** input/output.

ProgName (PNM)

An input/output used to enter the name of the program associated with the current ProgNumber. The name entered should be a valid file name for the PC3000 file system so that programs can be loaded from or saved to the file store.

Reset_Output (ROP)

The value that will appear on the **Output** when the **Mode** is set to **Reset.** Whilst the **Mode** remains in **Reset,** any change to **Reset_Output** is mirrored on **Output.**

Start_Mode (STM)

When a program is loaded, if **Mode** was **Run** at the start of the load operation, then following the load the mode will be set according to the **Start_Mode** of the newly loaded program. Possible values for **Start_Mode** are:

Reset(0):	After loading the program, Mode will be set to Reset.
Run(1):	After loading the program, Mode will be set to Run.
Hold(2):	After loading the program, Mode will be set to Hold.
Track(3):	After loading the program, Mode will be set to Track.
SkipSeg(4):	After loading the program, Mode will be set to SkipSeg.
NxtUpSeg(5):	After loading the program, Mode will be set to NxtUpSeg.
NxtDnSeg(6):	After loading the program, Mode will be set to NxtDnSeg.

However if the previous program had ended (i.e. **ProgramEnd** is Yes prior to load being selected), the new program will be loaded, and on completion of the load **Mode** will return to **Run,** but the program end flag will remain true and all outputs will remain unchanged from the values pertaining prior to the load being requested.

Process_Val (PV)

Normally the **Process_Val** of the PID loop being controlled by the Programmer function block would be wired to this input. This gives information to the function block that enables decisions to be taken based on the real world performance. **Output** can be set to track this input, or the ramp may be temporarily stopped if the process under control deviates from the setpoint by more than a given amount (this is know as "holdback"). It depends on the application whether a connection is needed to this input or not.

HB_Mode (HM)

By changing this input, the way in which holdback operates can be selected. Holdback operates during a ramp - there is no automatic hold-back during a dwell.

Off(0):	No holdback is in operation.

Lower(1): When **HB_Mode** is set to Lower, holdback will become active if **Process_Val** is less than or equal to **Output** minus the **HB_Deviation**. In other words, the Output will stop at its current position until **Process_Val** increases. The output **HB_Active** will also be true under these circumstances. This mode is usually used in cases where there is danger of the setpoint (i.e. the **Output** of the programmer function block) getting ahead of the process on rising ramps.

Upper(2): When **HB_Mode** is set to Upper, holdback will become active if **Process_Val** is greater than or equal to **Output** plus the **HB_Deviation.** In other words, the **Output** will stop at its current position until **Process_Val** decreases. The output **HB_Active** will also be true under these circumstances. This mode is usually used in cases where there is danger of the setpoint (i.e. the **Output** of the programmer function block) getting ahead of the process on falling ramps.

Band(3): If the absolute difference between **Process_Val** and **Output i**s greater than or equal to **HB_Deviation,** holdback will become active if **HB_Mode** is set to Band. In other words, the **Output** will stop at its current position until this condition is no longer true. The output **HB_Active** will also be true under these circumstances. This mode is used to protect the process from lagging too far behind the setpoint (i.e. the **Output** of the programmer function block) on both rising and falling ramps.

HB_Deviation (HD)

Used in conjunction with **Process_Val** and **Output** to determine whether holdback should become active, depending on the setting of **HB_Mode.** It defines limits relative to **Output** within which **Process_Val** is expected to remain.

End_Segment (ES)

The last segment to be considered as part of the profile. This allows a profile to be brought to an end before all non-null segments have been executed. A change made to this parameter whilst a program is running will become effective if the segment made to be the end segment has not yet completed executing.

Num_Loops (NL)

The number of times the current profile will be repeated before passing to the next program or flagging that the program has ended if **NextProgNum** is None.

NextProgNum (NPN)

This input allows a new program to be loaded from the file store once the thecurrent program has completed.

None(0): No program will be loaded when the current program completes.

Prog_1(1): The program whose name is given as **ProgName** when **ProgNumber**=1 will be loaded from the PC3000 File Store when the current program completes. If there is no ProgName associated with **ProgNumber**=1 then the ProgStatus will show **LoadErr** and **Mode** will go to **Reset.**

Prog_2(2): The program whose name is given as **ProgName** when **ProgNumber**=2 will be loaded from the PC3000 File Store when the current program completes. If there is no **ProgName** associated with **ProgNumber**=2 then the **ProgStatus** will show **LoadErr** and **Mode** will go to **Reset.**

"

etc up to

Prog_32(32)

RampTime1 (RT1)

The time that will be required for **Output** to move from its current value to the ramp level of segment 1 in standard PC3000 Time format.

RampDO1 (DD1)

This gives the possibility of setting each of the eight digital outputs of the Prog8Rate function block to a particular state during the ramp of segment 1. The value of this variable is interpreted as an eight bit pattern, the lowest bit setting **Dig_Out_1,** and the highest bit setting **Dig_Out_8.**

RampLvl1 (RL1)

The level of **Output** which, when achieved, will cause the dwell of segment 1 to commence.

DwellTime1 (DT1)

The length of time for which the **Output** will remain static at **RampLvl1** after the ramp part of segment 1 is completed.

DwellDO1 (DD1)

This gives the possibility of setting each of the eight digital outputs of the **Prog8Time**function block to a particular state during the dwell of segment 1. The value of this variable is interpreted as an eight bit pattern, the lowest bit setting **Dig_Out_1,** and the highest bit setting **Dig_Out_8.**

RampTime2 (RT2)

The time that will be required for **Output** to move from its current value to the ramp level of segment 2 in standard PC3000 Time format.

RampDO2 (RD2)

This gives the possibility of setting each of the eight digital outputs of the **Prog8Time** function block to a particular state during the ramp of segment 2. The value of this variable is interpreted as an eight bit pattern, the lowest bit setting **Dig_Out_1,** and the highest bit setting **Dig_Out_8.**

RampLvl2 (RL2)

The level of **Output** which, when achieved, will cause the dwell of segment 2 to commence.

DwellTime2 (DT2)

The length of time for which the **Output** will remain static at **RampLvl1** after the ramp part of segment 2 is completed.

DwellDO2 (DD2)

This gives the possibility of setting each of the eight digital outputs of the Prog8Rate function block to a particular state during the dwell of segment 2. The value of this variable is interpreted as an eight bit pattern, the lowest bit setting **Dig_Out_1,** and the highest bit setting **Dig_Out_8.**

"

up to

DwellDO8 (DD8)

CommsSegNum

Indication of which Rate, Level and Dwell the communications mnemonics are using.

See Edit Segment Number in Table 15-5.

Output (OP)

The current value of the profile being produced by the program now running. This **Output** would normally be wired to the **Setpoint** input of a PID loop or used in some other way as a setpoint for control action.

HB_Active (HA)

If **HB_Active** is true this indicates that the value currently being fed to the **Process_Val** input deviates from the current value of **Output** by more than the amount defined by **HB_Mode** and **HB_Deviation.**

ProgramEnd (PE)

This output indicates when the current program has ended, or where multiple programs are chained together using **NextProgNum,** when the last program in a chain has ended.

Status (SS)

The success or otherwise of the last file operation to or from the PC3000 File Store is indicated by this output, or, during a file operation, the operation in progress is indicated.

Ok(0):	The last file store operation was succesful, either save or load.
Saving(1):	A save operation is in progress to the file store .
Loading(2):	A load operation is in progress from the file store.
SaveErr(3):	The last save to file store operation failed. This could be for a variety of reasons, such as no file store exists, the file store is full or the file name specified in the **ProgName** parameter is invalid.
LoadErr(4):	The last load from file store operation failed. This could be for a variety of reasons, such as no file store exists or the file name specified in the **ProgName** parameter is invalid or is not a recipe file.This error will also be reported if a program saved by a Prog8Rate function block is loaded.

LoopsRemain (TR)

The number of loops (repeats) of the current program that have yet to be run once the current loop is completed.

CurrentSeg (CS)

The number of the segment of the current program currently being executed.This could either be the ramp or dwell part of the segment.

CurrentMode (CM)

This indicates whether the Ramp part or the Dwell part of the current segment is being executed. Normally the ramp part of a segent is followed by the dwell, and then the next segment begins to execute.

However, if **Mode** is set to Track when the dwell has already commenced, the **CurrentMode** will revert to Ramp. When **Mode** is set to Run again, the Output will ramp to the current segment **RampLvl** value in the current segment **RampTime** and then **CurrentMode** will again indicate Dwell. This second dwell period will last for the full **DwellTime** of the current segment.

CurrentTmRem (CTR)

The time that will be required to complete the operation indicated by **CurrentMode** (i.e. a ramp or dwell) assuming their are no interruptions caused by hold back becoming active, or due to **Mode** being other than Run. The calculations are based on the current value of **Output** and the profile parameters for the current segment. If hold back becomes active, or **Mode** is changed, **CurrentTmRem** indicates the time required to complete the operation indicated by **CurrentMode** if hold back were to become immediately inactive, or **Mode** were to be immediately changed to Run.

SegTmRem (STR)

The time that will be required to complete the current segment assuming their are no interruptions caused by hold back becoming active, or due to **Mode** being other than Run. The calculations are based on the current value of **Output** and the profile parameters for the current segment.

If hold back becomes active, or **Mode** is changed, **SegTmRem** indicates the time required to complete the current segment if hold back were to become immediately inactive, or **Mode** were to be immediately changed to Run.

ProgTmRem (PTR)

The time that will be required to complete the remaining number of loops of the current profile assuming their are no interruptions caused by hold back becoming active, or due to **Mode** being other than Run. The calculations are based on the current value of **Output** and the profile parameters for the current segment together with the **Num_Loops** parameter.

If hold back becomes active, or **Mode** is changed, **ProgTmRem** indicates the time required to complete the current segment if hold back were to become immediately inactive, or Mode were to be immediately changed to Run.

When Mode is set to Run and changes are made to program parameters, **ProgTmRem** will only be recalculated at the start of the next segment or at the changeover from a ramp to a dwell.

Dig_Out_1 (DO1)

A digital output which is driven by bit 0 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 0 of **RampDO4** is 1 and bit 0 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_1** will be switched on, and during the dwell in segment 4 **Dig_Out_1** will be switched off.

Dig_Out_2 (DO2)

A digital output which is driven by bit 1 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 1 of **RampDO4** is 1 and bit 1 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_2** will be switched on, and during the dwell in segment 4 **Dig_Out_2** will be switched off.

Dig_Out_3 (DO3)

A digital output which is driven by bit 2 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 2 of **RampDO4** is 1 and bit 2 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_3** will be switched on, and during the dwell in segment 4 **Dig_Out_3** will be switched off.

Dig_Out_4 (DO4)

A digital output which is driven by bit 3 of the **RampDO** and **DwellD** input/outputs of the current segment. For example, if bit 3 of **RampDO4** is 1 and bit 3 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_4** will be switched on, and during the dwell in segment 4 **Dig_Out_4** will be switched off.

Dig_Out_5 (DO5)

A digital output which is driven by bit 4 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 4 of **RampDO4** is 1 and bit 4 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_5** will be switched on, and during the dwell in segment 4 **Dig_Out_5** will be switched off.

Dig_Out_6 (DO6)

A digital output which is driven by bit 5 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 5 of **RampDO4** is 1 and bit 5 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_6** will be switched on, and during the dwell in segment 4 **Dig_Out_6** will be switched off.

Dig_Out_7 (DO7)

A digital output which is driven by bit 6 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 6 of **RampDO4** is 1 and bit 6 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_7** will be switched on, and during the dwell in segment 4 **Dig_Out_7** will be switched off.

Dig_Out_8 (DO8)

A digital output which is driven by bit 7 of the **RampDO** and **DwellDO** input/outputs of the current segment. For example, if bit 7 of **RampDO4** is 1 and bit 7 of **DwellDO4** is 0, then during the ramp in segment 4 **Dig_Out_8** will be switched on, and during the dwell in segment 4 **Dig_Out_8** will be switched off.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Mode	**ENUM**	Reset (0)	Oper	Oper	See Parameter descriptions	
Address	**STR**	'EPP1'	Super	Super	Max 4 characters	
RealFormat	**STR**	''	Super	Super	Single character	

WriteInhibit	**BOOL**	Rd_Wr(0)	Super	Super	Senses	Rd_Wr(0)
						Rd_Only(1)

ProgNumber	**DINT**	1	Oper	Oper	High Limit Low Limit	32 1
ProgName	**STR**	''	Oper	Oper	Max 12 characters	
Reset_Output	**REAL**	0	Super	Super	High Limit Low Limit	+3.402823E+38 -3.402823E+38
Start_Mode	**ENUM**	Run (1)	Oper	Oper	See parameter list	
Process_Val	**REAL**	0	Oper	Oper	High Limit Low Limit	+3.402823E+38 -3.402823E+38
HB_Mode	**ENUM**	Off (0)	Oper	Oper	See parameter list	
HB_Deviation	**REAL**	0	Oper	Oper	High Limit Low Limit	+3.402823E+38 -3.402823E+38
End_Segment	**DINT**	8	Super	Super	High Limit Low Limit	8 1
Num_Loops	**DINT**	1	Super	Super	High Limit Low Limit	999 0ms
NextProgNum	**ENUM**	None(0)	Super	Super	See parameter list	
RampTime1 to RampTime8	**REAL**	0	Super	Super	High Limit Low Limit	23d23h59m59s999ms 0ms
RampDO1 to RampDO8	**DINT**	0	Super	Super	High Limit Low Limit	256 0
RampLv11 to RampLv18	**REAL**	0	Super	Super	High Limit Low Limit	+3.402823E+38 -3.402823E+38
DwellTime1 to DwellTime8	**TIME**	0ms	Super	Super	High Limit Low Limit	23d23h59m59s999ms 0ms
DwellD01-to DwellD08	**DINT**	0	Super	Super	High Limit Low Limit	256 0
CommsSegNum	**DINT**	1	Super	Super	High Limit Low Limit	8 1
Output	**REAL**	0	Oper	Block	High Limit Low Limit	+3.402823E+38 -3.402823E+38

Table 15-6 Prog8Time Parameter Attributes (continued)

Name	**Type**	**Cold Start**	**Read Access**	**Write Access**	**Type Specific Information**	

HB_Active	**BOOL**	No(0)	Oper	Block	Senses	No(0)
						Yes(1)

ProgramEnd	**BOOL**	Yes(1)	Oper	Block	Senses	No(0)
						Yes(1)

Status	**ENUM**	Ok(0)	Oper	Block	See parameter list	
LoopsRemain	**DINT**	0	Oper	Block	High Limit Low Limit	999 0
CurrentSeg	**DINT**	8	Oper	Block	High Limit Low Limit	8 1

CurrentMode	**BOOL**	Dwell(1)	Oper	Block	Senses	Ramp(0)
						Dwell(1)

CurrentTmRem	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms
SegTmRem	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms
ProgTmRem	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms

Dig_Out_1 to Dig_Out_8	**BOOL**	Off(0)	Oper	Block	Senses	Off(0)
						On(1)

Table 15-5 Prog8Time Parameter Attributes (continued)