UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS ELECTRÓNICOS

ALGORITMO EXPERTO PARA LA OPTIMIZACIÓN DE LA COMPENSACIÓN SELECTIVA DE UN FILTRO ACTIVO DE POTENCIA CONECTADO EN EL EDIFICIO 8G DE LA UPV

TRABAJO DE FIN DE MÁSTER

Autor:

Daniel Engiberto Granda Gutiérrez

Dirigida por:

Dr. D. Francisco J. Gimeno Sales

Dr. D. Salvador Orts Grau

Valencia, 2016

ÍNDICE DE CONTENIDO DE LOS ANEXOS

Contenido:

ANEXOS

ANEXO I

1. PARÁMETROS AHF SHAFFNER ECOSINE ACTIVE *FN 3430-60-400-4*

Grupo de Parámetros	Significado	Comentarios
P0xx	Datos de dispositivo	**Solo lectura** Muestra los datos del dispositivo (corriente nominal, corriente de sobrecarga,..)
P1xx	Valores Medidos	Solo Lectura Muestra los valores medidos (Voltajes principales, corrientes de carga, corrientes principales, corrientes del filtro,)
P2xx	Configuración básica	**Parámetro de puesta en marcha** (configuración del lenguaje, fecha, etc.)
P3xx	Transformador de corriente	**Parámetro de puesta en marcha** (ajuste para la posición de transformador de corriente, tasa de transformacion , funcionamiento en paralelo de ECOsine® active , ...)
P4xx	Tipo de compensación	**Parámetro de puesta en marcha** (Activación de compensación de potencia reactiva, compensación de corriente armónica)
P6xx	Alarmas	**Sólo lectura** Muestra los mensajes de error

Tabla 1.1. Grupos de parámetros del Filtro activo Shaffner Ecosine Active *FN 3430-60-400-4*

No.	Designación	Unidades	Significado
120	Corriente de línea rms L1	A	Corriente de Red L1 - rms
121	Corriente de línea rms L2	A	Corriente de Red L2 - rms
122	Corriente de línea rms L3	A	Corriente de Red L3 - rms
133	Corriente de carga rms L1	A	Corriente de Carga L1 - rms
134	Corriente de carga rms L2	A	Corriente de Carga L2 - rms
135	Corriente de carga rms L3	A	Corriente de Carga L3 - rms
140	Corriente de salida rms L1	A	Corriente de Compensación L1 - rms
141	Corriente de salida rms L2	A	Corriente de Compensación L2 - rms
142	Corriente de salida rms L3	A	Corriente de Compensación L3 - rms

Tabla 1.2. Grupos de parámetrosP1xx(solo lectura) del Filtro activo Shaffner Ecosine Active *FN 3430-60-400-4*

No.	Designación	Configuración de Fábrica	Significado
405	Balanceo de Carga	OFF	Activación o desactivación del balanceo de carga entre fases | OFF | ON El balanceo de carga está activo **solo si simultáneamente el parámetro P410 = ON**.
407	Prioridad a Plena Carga	0 = no prioridad	Compensación de prioridad a plena carga. **0 = ninguna** (limitación simétrica de compensación de armónicos y reactiva - valor por defecto) **1 = Corriente Reactiva** (Prioridad para compensar corriente reactiva a plena carga) **2 = Armónicos** (prioridad para compensar armónicos a plena carga)
410	Compensación Armónica	OFF	Activación o desactivación de la compensación de corrientes armónicas **OFF** Los controladores de compensación armónica (P411 - P434) no están activos **ON** Los controladores de compensación armónica (P411 - P434) están activos

Tabla 1.3. Grupos de parámetrosP4xx (Solo lectura) del Filtro activo Shaffner Ecosine Active *FN 3430-60-400-4*

Nro. de Parámetro	Designación	Configuración de fábrica	Significado
411	Compensación A3	80%	Grado de compensación ajustable del 3er armónico 1%....100%
412	Compensación A5	80%	Grado de compensación ajustable del 5to armónico 1%....100%
413	Compensación A7	80%	Grado de compensación ajustable del 7mo armónico 1%....100%
414	Compensación A9	50%	Grado de compensación ajustable del 9no armónico 1%....100%
415	Compensación A11	50%	Grado de compensación ajustable del 11vo armónico 1%....100%
416	Compensación A13	40%	Grado de compensación ajustable del 13vo armónico 1%....100%
417	Compensación A15	0%	Grado de compensación ajustable del 15vo armónico 1%....100%
418	Compensación A17	30%	Grado de compensación ajustable del 17vo armónico 1%....100%
419	Compensación A19	20%	Grado de compensación ajustable del 19vo armónico 1%....100%
420	Compensación A21	0%	Grado de compensación ajustable del 21vo armónico 1%....100%

Tabla 1.4. Grupos de parámetrosP4xx (Configuración del grado de compensación) del Filtro activo Shaffner Ecosine Active *FN 3430-60-400-4*

Mensaje en Display	Codigo (P20)	Significado	Nota
Fault	0	Error, ECOsine active no está activo	Mirar códigos de Error
	1		No se usa
Operation	2	ECOsine active operando	-
Off	3	ECOsine active está apagado	
Full Load	4	ECOsine active operando a plena carga	
DC link too high	5	Desconexión temporal del ECOsine active debido al exceso de voltaje máximo del enlace DC.	La cusa puede ser rápidas fluctuaciones de carga
Sobrecorriente	6	Desconexión temporal del ECOsine active debido al exceso de la corriente máxima de salida	
Standby	7	Dispositivo apagado, debido a que la corriente de compensación requerida es menor que el límite de espera.	
Line synchronization	8	Después de "Prender la fuente de Alimentación" ECOsine active está a la espera para sincronizarse exitosamente con la red. ECOsine active no está en operación	
	9		No se usa
Red operation	10	El control interno ha desactivado los controladores de los armónicos más altos, la operación del ECOsine active continua.	
Check CT	11	ECOsine active no se ha encendido todavía y la señal de medida de los transformadores de corriente no es recomendable (Posible detección de carga no se previene)	
THDu resonance	12	Apago temporal del ECOsine active para comprobar la resonancia de línea.	

Tabla 1.5. Mensajes de estado del Filtro activo Shaffner Ecosine Active *FN 3430-60-400-4*

Parameter No	Modbus-address	Name	Access/ function code	Datatype	Unit	Values
2	20	rated current	READ_ONLY	UnsignedInteger_16	A	
3	30	overload current	READ_ONLY	UnsignedInteger_16	A	
8	80	MAC address	READ_ONLY	string		example: 08:00:70:22:44:11 = 17 ASCII characters
9	90	serial no of control board SE0601	READ_ONLY	UnsignedInteger_32		
10	100	firmware version	READ_ONLY	string		example: V02.06.30 = 9 ASCII characters
11	110	firmware LT controller	READ_ONLY	UnsignedInteger_8		
12	120	status LT controller	READ_ONLY	UnsignedInteger_8		0 = (not used) 1 =precharging 2 = LTC ready for operation 3 = (not used) 4 = error power stack
13	130	LTC fail code	READ_ONLY	UnsignedInteger_16		0 ... 65535
14	140	LTC fail code	READ_ONLY	UnsignedInteger_8		0 ... 255
15	150	version device coding	READ_ONLY	UnsignedInteger_8		0 ... 255
16	160	CT offset secondary side L1	READ_ONLY	SinglePrecisionFloatingPoint	mA	
17	170	CT offset secondary side L2	READ_ONLY	SinglePrecisionFloatingPoint	mA	
18	180	CT offset secondary side L3	READ_ONLY	SinglePrecisionFloatingPoint	mA	
20	200	state	READ_ONLY	UnsignedInteger_16		0 = failure 1 = (not used) 2 = operation 3 = off 4 = full load 5 = DC link to high 6 = over current 7 = standby 8 = line synchronization 9 = (not used) 10= red.operation 11= check CT 12= THDu reson

21	210	cause of fault	READ_ONLY	UnsignedInteger_16		0 = no failure 1 = overcurrent 2 = fan error 3 = power failure 4 = (not used) 5 = CPU error 6 = FPGA error 7 = overtemperature 8 = (not used) 9 = liquid cooling 10 = IGBT Uce max 11 = harm. Reson. 12 = contr. Instable 13 = notch filter 14 = THDu reson. 15 = CPU Overtemp
30	300	operatinghours	READ_ONLY	SinglePrecisionFloatingPoint	x.xxx h	
100	1000	mains frequency	READ_ONLY	SinglePrecisionFloatingPoint	Hz	
101	1010	THDi	READ_ONLY	SinglePrecisionFloatingPoint	%	
102	1020	Power factor	READ_ONLY	SinglePrecisionFloatingPoint		
103	1030	DC link voltage	READ_ONLY	SinglePrecisionFloatingPoint	V	
104	1040	device load	READ_ONLY	SinglePrecisionFloatingPoint	%	
105	1050	active power L1	READ_ONLY	SinglePrecisionFloatingPoint	kW	
106	1060	active power L2	READ_ONLY	SinglePrecisionFloatingPoint	kW	
107	1070	active power L3	READ_ONLY	SinglePrecisionFloatingPoint	kW	
108	1080	output reactive current rms of active filter	READ_ONLY	SinglePrecisionFloatingPoint	A	
109	1090	direction of rotation	READ_ONLY	UnsignedInteger_16		0 = clockwise 1 = counter clockwise 2 = no synchronization
110	1100	line volt. rms U12	READ_ONLY	SinglePrecisionFloatingPoint	V	
111	1110	line volt. rms U23	READ_ONLY	SinglePrecisionFloatingPoint	V	
112	1120	line volt. rms U31	READ_ONLY	SinglePrecisionFloatingPoint	V	
113	1130	line voltage U1	READ_ONLY	SinglePrecisionFloatingPoint	V	
114	1140	line voltage U2	READ_ONLY	SinglePrecisionFloatingPoint	V	

115	1150	line voltage U3	READ_ONLY	SinglePrecisionFloatingPoint	V	
116	1160	line voltage U12	READ_ONLY	SinglePrecisionFloatingPoint	V	
117	1170	line voltage U23	READ_ONLY	SinglePrecisionFloatingPoint	V	
118	1180	line voltage U31	READ_ONLY	SinglePrecisionFloatingPoint	V	
119	1190	THDu	READ_ONLY	SinglePrecisionFloatingPoint	V	
120	1200	current rms L1	READ_ONLY	SinglePrecisionFloatingPoint	A	
121	1210	current rms L2	READ_ONLY	SinglePrecisionFloatingPoint	A	
122	1220	current rms L3	READ_ONLY	SinglePrecisionFloatingPoint	A	
123	1230	line current L1	READ_ONLY	SinglePrecisionFloatingPoint	A	
124	1240	line current L2	READ_ONLY	SinglePrecisionFloatingPoint	A	
125	1250	line current L3	READ_ONLY	SinglePrecisionFloatingPoint	A	
126	1260	neutral lline current L3	READ_ONLY	SinglePrecisionFloatingPoint	A	
130	1300	loadcur. rms L1	READ_ONLY	SinglePrecisionFloatingPoint	A	
131	1310	loadcur. rms L2	READ_ONLY	SinglePrecisionFloatingPoint	A	
132	1320	loadcur. rms L3	READ_ONLY	SinglePrecisionFloatingPoint	A	
133	1330	loadcurrent L1	READ_ONLY	SinglePrecisionFloatingPoint	A	
134	1340	loadcurrent L2	READ_ONLY	SinglePrecisionFloatingPoint	A	
135	1350	loadcurrent L3	READ_ONLY	SinglePrecisionFloatingPoint	A	
136	1360	neutral loadcurrent	READ_ONLY	SinglePrecisionFloatingPoint	A	
140	1400	outcurrent rms L1	READ_ONLY	SinglePrecisionFloatingPoint	A	
141	1410	outcurrent rms L2	READ_ONLY	SinglePrecisionFloatingPoint	A	
14	1420	outcurrent	READ_O	SinglePrecisionFlo	A	

2		rms L3	NLY	atingPoint		
143	1430	outcurrent L1	READ_ONLY	SinglePrecisionFloatingPoint	A	
144	1440	outcurrent L2	READ_ONLY	SinglePrecisionFloatingPoint	A	
145	1450	outcurrent L3	READ_ONLY	SinglePrecisionFloatingPoint	A	
146	1460	neutral outcurrent	READ_ONLY	SinglePrecisionFloatingPoint	A	
147	1470	neutral line current rms	READ_ONLY	SinglePrecisionFloatingPoint	A	
148	1480	neutral load current rms	READ_ONLY	SinglePrecisionFloatingPoint	A	
149	1490	neutral output current rms	READ_ONLY	SinglePrecisionFloatingPoint	A	
177	1770	max. harmonic	READ_ONLY	UnsignedInteger_16		
178	1780	THDu refer.	READ_ONLY	SinglePrecisionFloatingPoint	%	
179	1790	THDu limit	READ_ONLY	SinglePrecisionFloatingPoint	%	
180	1800	heatsink temper.	READ_ONLY	UnsignedInteger_8	degr. C	
181	1810	internal temperature	READ_ONLY	UnsignedInteger_8	degr. C	
182	1820	fan 3 speed	READ_ONLY	UnsignedInteger_16	rpm	
183	1830	fan 4 speed	READ_ONLY	UnsignedInteger_16	rpm	
184	1840	harm. controller peak	READ_ONLY	SinglePrecisionFloatingPoint	V	
185	1850	propor. controller rms	READ_ONLY	SinglePrecisionFloatingPoint	V	
186	1860	notch filter rms	READ_ONLY	SinglePrecisionFloatingPoint	A	
187	1870	p-contr. rms max.	READ_ONLY	SinglePrecisionFloatingPoint	V	
200	2000	language	READ_WRITE	UnsignedInteger_16		0 = german 1 = english
201	2010	Polarity alarm output	READ_WRITE	UnsignedInteger_16		0 = low active 1 = high active
(20	2020	Switch on	READ_W	UnsignedInteger_1		0 = terminal strip 1 = direct ON

2			RITE	6		2 = direct OFF
210	2100	load default values	READ_WRITE	UnsignedInteger_16		0 = no action 1 = load factory settings
220	2200	Date	READ_WRITE	Date		
230	2300	Modbus node address	READ_WRITE	UnsignedInteger_16		1 ... 255
231	2310	Modbus baudrate	READ_WRITE	UnsignedInteger_16		0 = 2400 1 = 9600 2 = 14400 3 = 19200 4 = 38400 5 = 57600 6 = 64800 7 = 115200
232	2320	Modbus parity	READ_WRITE	UnsignedInteger_16		0 = NO 1 = ODD 2 = EVEN
233	2330	Modbus stop bit	READ_WRITE	UnsignedInteger_16		0 ... 2
240	2400	IP address	READ_WRITE	UnsignedInteger_32		example 0XC0A80102 = 192.168.1.2
241	2410	DHCP	READ_WRITE	UnsignedInteger_16		0 = DHCP off 1 = DHCP ON
242	2420	subnet mask	READ_WRITE	UnsignedInteger_32		example 0XFFFFFF00 = 255.255.255. 0
243	2430	default gateway	READ_WRITE	UnsignedInteger_32		example 0XC0A80132 = 192.168.1.50
300	3000	transf. placement	READ_WRITE	UnsignedInteger_16		0 = mains side 1 = load side
310	3100	transformer ratio	READ_WRITE	UnsignedInteger_16	: 5A	50 ... 5000
311	3110	CT check	READ_WRITE	UnsignedInteger_16		0 = OFF 1 = ON
320	3200	total current parallel	READ_WRITE	UnsignedInteger_16	A	
400	4000	reactive power	READ_WRITE	SinglePrecisionFloatingPoint	%	0 ... 100%
401	4010	Setpoint displace-ment power factor	READ_WRITE	SinglePrecisionFloatingPoint		0,0 ... 1,0
405	4050	Load balancing	READ_WRITE	UnsignedInteger_16		0 = ON 1 = OFF (From V02.06.XX)
406	4060	Standby threshold	READ_WRITE	UnsignedInteger_16	%	0 100%
407	4070	priority full load	READ_WRITE	UnsignedInteger_16		0 = NONE 1 = reactive current 2 = harmonics
41	4100	Harmo nic	READ_W	UnsignedInteger_1		0 = off

0		compe ns.	RITE	6		1 = on
411	4110	h3 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
412	4120	h5 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
413	4130	h7 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
414	4140	h9 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
415	4150	h11 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
416	4160	h13 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
417	4170	h15 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
418	4180	h17 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
419	4190	h19 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
420	4200	h21 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
421	4210	h23 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
422	4220	h25 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
423	4230	h27 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
424	4240	h29 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
425	4250	H31 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
426	4260	H33 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
427	4270	H35 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	

428	4280	H37 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
429	4290	H39 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
430	4300	H41 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
431	4310	H43 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
432	4320	H45 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
433	4330	H47 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	
434	4340	H49 degree of compensation	READ_WRITE	SinglePrecisionFloatingPoint	%	

Parameter No	Modbus-address	Name	Access/ function code	Datatype	Unit	Values
700	7000	FFT selection	READ_WRITE	UnsignedInteger_16		0 = iLine_L1 1 = iLine_L2 2 = iLine_L3 3 = iLine_neutral 4 = iOut_L1 5 = iOut_L2 6 = iOut_L3 7 = iOut_neutral for the selected value a FFT analysis is done and available in parameters 701ff
701	7010	FFT fundamental peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
702	7020	FFT harm2 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
703	7030	FFT harm3 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
704	7040	FFT harm4 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
705	7050	FFT harm5 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	

706	7060	FFT harm6 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
707	7070	FFT harm7 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
708	7080	FFT harm8 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
709	7090	FFT harm9 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
710	7100	FFT harm10 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
711	7110	FFT harm11 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
712	7120	FFT harm12 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
713	7130	FFT harm13 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
714	7140	FFT harm14 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
715	7150	FFT harm15 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
716	7160	FFT harm16 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
717	7170	FFT harm17 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
718	7180	FFT harm18 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
719	7190	FFT harm19 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
720	7200	FFT harm20 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
721	7210	FFT harm21 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
722	7220	FFT harm22	READ_ONLY	SinglePrecisionFloatingPoint	A	

		peak				
723	7230	FFT harm23 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
724	7240	FFT harm24 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
725	7250	FFT harm25 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
726	7260	FFT harm26 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
727	7270	FFT harm27 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
728	7280	FFT harm28 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
729	7290	FFT harm29 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
730	7300	FFT harm30 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
731	7310	FFT harm31 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
732	7320	FFT harm32 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
733	7330	FFT harm33 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
734	7340	FFT harm34 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
735	7350	FFT harm35 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
736	7360	FFT harm36 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
737	7373	FFT harm37 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	

738	7380	FFT harm38 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
739	7390	FFT harm39 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
740	7400	FFT harm40 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
741	7410	FFT harm41 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
742	7420	FFT harm42 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
743	7430	FFT harm43 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
744	7440	FFT harm44 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
745	7450	FFT harm45 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
746	7460	FFT harm46 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
747	7474	FFT harm47 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
748	7480	FFT harm48 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	
749	7490	FFT harm49 peak	READ_ONLY	SinglePrecisionFloatingPoint	A	

Tabla 1.6. Direcciones Modbus para la comunicación mediante el protocolo Modbus del Filtro activo Shaffner Ecosine Active *FN 3430-60-400-4*

ANEXO II

2. CÓDIGO EN MATLAB (SIMULACIÓN)

```
function [sys,x0,str,ts] = Algoritmo_Iref_APF_Daniel(t,x,u,flag,xFFm)

global val_ant_k_1;
global val_act_k;
global xnnx;
global xTTm;
global xyy;
global sp;
global sn;
    global primeravez;
    global GlobalM;
    global GlobalPh;
    global Umbral;
    global Umbral_Desq;
    global Global_IR_ca;
    global Global_IS_ca;
    global Global_IT_ca;
    global Global_IR_ca_reactiva;
    global Global_IS_ca_reactiva;
    global Global_IT_ca_reactiva;
```

```
21.     global Global_escalado_react;
22.
23.     %%%%%% Variables de Salida....
24.      global I_R_Ref_COMPENSA;
25.      global I_S_Ref_COMPENSA;
26.      global I_T_Ref_COMPENSA;
27.      global senal;
28.      global factcmp_reord;
29.      global FacUtil_F_R;
30.      global FacUtil_F_S;
31.      global FacUtil_F_T;
32.      global FacUtil_Total;
33.
34.     %%%%%% Variables de Entrada....
35.     global I_MAXIMA_COMPENSA;
36.     global NUM_ALGORITMO;
37.
38.     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39.     % FASE TRIFASICA-R
40.         %MAGNITUD-----------FASE
41.     % Armonico-1-Fundamental
42.     global IR_1_Mag; global IR_1_Fase;
43.
44.     % Armonico-3º
45.     global IR_3_Mag; global IR_3_Fase;
46.     % Armonico-5º
47.     global IR_5_Mag; global IR_5_Fase;
48.     % Armonico-7º
49.     global IR_7_Mag; global IR_7_Fase;
50.     % Armonico-9º
51.     global IR_9_Mag; global IR_9_Fase;
52.     % Armonico-11º
53.     global IR_11_Mag; global IR_11_Fase;
54.     % Armonico-13º
55.     global IR_13_Mag; global IR_13_Fase;
56.     % Armonico-15º
57.     global IR_15_Mag; global IR_15_Fase;
58.     % Armonico-17º
59.     global IR_17_Mag; global IR_17_Fase;
60.     % Armonico-19º
61.     global IR_19_Mag; global IR_19_Fase;
62.     % Armonico-21º
63.     global IR_21_Mag; global IR_21_Fase;
64.
65.     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
66.     % FASE TRIFASICA-S
67.         %MAGNITUD-----------FASE
68.     % Armonico-1-Fundamental
69.     global IS_1_Mag; global IS_1_Fase;
70.     % Armonico-3º
71.     global IS_3_Mag; global IS_3_Fase;
72.     % Armonico-5º
73.     global IS_5_Mag; global IS_5_Fase;
74.     % Armonico-7º
75.     global IS_7_Mag; global IS_7_Fase;
76.     % Armonico-9º
```

```
77.     global IS_9_Mag; global IS_9_Fase;
78.     % Armonico-11º
79.     global IS_11_Mag; global IS_11_Fase;
80.     % Armonico-13º
81.     global IS_13_Mag; global IS_13_Fase;
82.     % Armonico-15º
83.     global IS_15_Mag; global IS_15_Fase;
84.     % Armonico-17º
85.     global IS_17_Mag; global IS_17_Fase;
86.     % Armonico-19º
87.     global IS_19_Mag; global IS_19_Fase;
88.     % Armonico-21º
89.     global IS_21_Mag; global IS_21_Fase;
90.
91.     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
92.     % FASE TRIFASICA-T
93.         %MAGNITUD-----------FASE
94.     % Armonico-1-Fundamental
95.     global IT_1_Mag; global IT_1_Fase;
96.     % Armonico-3º
97.     global IT_3_Mag; global IT_3_Fase;
98.     % Armonico-5º
99.     global IT_5_Mag; global IT_5_Fase;
100.    % Armonico-7º
101.    global IT_7_Mag; global IT_7_Fase;
102.    % Armonico-9º
103.    global IT_9_Mag; global IT_9_Fase;
104.    % Armonico-11º
105.    global IT_11_Mag; global IT_11_Fase;
106.    % Armonico-13º
107.    global IT_13_Mag; global IT_13_Fase;
108.    % Armonico-15º
109.    global IT_15_Mag; global IT_15_Fase;
110.    % Armonico-17º
111.    global IT_17_Mag; global IT_17_Fase;
112.    % Armonico-19º
113.    global IT_19_Mag; global IT_19_Fase;
114.    % Armonico-11º
115.    global IT_21_Mag; global IT_21_Fase;
116.
117.    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118.    % I RST
119.        %MAGNITUD-----------FASE
120.    % Secuencia-pos-Fundamental
121.    global Isp_Mag; global Isp_Fase;
122.    % Secuencia-neg-Fundamental
123.    global Isn_Mag; global Isn_Fase;
124.    % Secuencia-cero-Fundamental
125.    global Is0_Mag; global Is0_Fase;
126.
127.    %DECLARACION MIS VARIABLES
128.    global I_ref;
129.    global factcmp;
130.    %global factcmp_reord;
131.    %global Ies_REFERENCIA;
132.    %global i;
```

```
133.    global tmodulador;
134.    global I_MAXIMA_NUEVA;
135.    %=============================================================
136.    % CONTROL DEL SISTEMA Discretizado
137.    %=============================================================
138.    if flag == 0
139.
140.    sizes = simsizes;
141.
142.    sizes.NumContStates  = 0;
143.    sizes.NumDiscStates  = 0;
144.    sizes.NumOutputs     = 60;%12_FacComp, 12_I_R_COMP, 12_I_S_COMP,
   12_I_T_COMP, 12_Senal
145.    sizes.NumInputs      = 74% 22_AF_R, 22_AF_S, 22_AF_T, 6_s+-0,
   1_I_MAXIMA, 1_ALGORITMO
146.    sizes.DirFeedthrough = 1;
147.    sizes.NumSampleTimes = 1;   % at least one sample time is needed
148.
149.    sys = simsizes(sizes);
150.    x0  = [];
151.    str = [];
152.    ts  = [0];
153.
154.    val_ant_k_1=0;
155.    val_act_k=0;
156.    xnnx=0;
157.    tmodulador=0;
158.    xTTm=1/xFFm;
159.    sp=0;
160.    sn=0;
161.
162.    %%%%%% Variables de Salida....
163.    I_R_Ref_COMPENSA=zeros(12,1);
164.    I_S_Ref_COMPENSA=zeros(12,1);
165.    I_T_Ref_COMPENSA=zeros(12,1);
166.
167.    %%%%%% Variables de Entrada....
168.    %%%%%% INICIALIZAR VALORES....
169.    I_MAXIMA_COMPENSA=25;
170.    NUM_ALGORITMO=1;
171.
172.    % FASE TRIFASICA-R .....
173.    % Armonico-1-Fundamental
174.    IR_1_Mag=0; IR_1_Fase=0;
175.    % Armonico-3º
176.    IR_3_Mag=0; IR_3_Fase=0;
177.    % Armonico-5º
178.    IR_5_Mag=0; IR_5_Fase=0;
179.    % Armonico-7º
180.    IR_7_Mag=0; IR_7_Fase=0;
181.    % Armonico-9º
182.    IR_9_Mag=0; IR_9_Fase=0;
183.    % Armonico-11º
184.    IR_11_Mag=0; IR_11_Fase=0;
185.    % Armonico-13º
186.    IR_13_Mag=0; IR_13_Fase=0;
```

```
187.    % Armonico-15º
188.    IR_15_Mag=0; IR_15_Fase=0;
189.    % Armonico-17º
190.    IR_17_Mag=0; IR_17_Fase=0;
191.    % Armonico-19º
192.    IR_19_Mag=0; IR_19_Fase=0;
193.    % Armonico-21º
194.    IR_21_Mag=0; IR_21_Fase=0;
195.
196.    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
197.    % FASE TRIFASICA-S .....
198.    % Armonico-1-Fundamental
199.    IS_1_Mag=0; IS_1_Fase=0;
200.    % Armonico-3º
201.    IS_3_Mag=0; IS_3_Fase=0;
202.    % Armonico-5º
203.    IS_5_Mag=0; IS_5_Fase=0;
204.    % Armonico-7º
205.    IS_7_Mag=0; IS_7_Fase=0;
206.    % Armonico-9º
207.    IS_9_Mag=0; IS_9_Fase=0;
208.    % Armonico-11º
209.    IS_11_Mag=0; IS_11_Fase=0;
210.    % Armonico-13º
211.    IS_13_Mag=0; IS_13_Fase=0;
212.    % Armonico-15º
213.    IS_15_Mag=0; IS_15_Fase=0;
214.    % Armonico-17º
215.    IS_17_Mag=0; IS_17_Fase=0;
216.    % Armonico-19º
217.    IS_19_Mag=0; IS_19_Fase=0;
218.    % Armonico-21º
219.    IS_21_Mag=0; IS_21_Fase=0;
220.
221.    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
222.    % FASE TRIFASICA-T .....
223.    % Armonico-1-Fundamental
224.    IT_1_Mag=0; IT_1_Fase=0;
225.    % Armonico-3º
226.    IT_3_Mag=0; IT_3_Fase=0;
227.    % Armonico-5º
228.    IT_5_Mag=0; IT_5_Fase=0;
229.    % Armonico-7º
230.    IT_7_Mag=0; IT_7_Fase=0;
231.    % Armonico-9º
232.    IT_9_Mag=0; IT_9_Fase=0;
233.    % Armonico-11º
234.    IT_11_Mag=0; IT_11_Fase=0;
235.    % Armonico-13º
236.    IT_13_Mag=0; IT_13_Fase=0;
237.    % Armonico-15º
238.    IT_15_Mag=0; IT_15_Fase=0;
239.    % Armonico-17º
240.    IT_17_Mag=0; IT_17_Fase=0;
241.    % Armonico-19º
242.    IT_19_Mag=0; IT_19_Fase=0;
```

```
243.    % Armonico-21º
244.    IT_21_Mag=0; IT_21_Fase=0;
245.
246.    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
247.    % I RST .....
248.        %MAGNITUD-----------FASE
249.    % Secuencia-positiva-Fundamental
250.    Isp_Mag=0; Isp_Fase=0;
251.    % Secuencia-negativa-Fundamental
252.    Isn_Mag=0; Isn_Fase=0;
253.    % Secuencia-cero-Fundamental
254.    Is0_Mag=0; Is0_Fase=0;
255.
256.    %INICIALIZACION DE MIS VARIABLES
257.    I_ref=0;
258.    factcmp=zeros([12,1]);
259.    factcmp_reord=zeros([12,1]);
260.    senal=zeros([12,1]);
261.    Umbral=0.3;
262.    Umbral_Desq=0.1;
263.    %Ies_REFERENCIA=zeros([11,1]);
264.    %i=0;
265.    primeravez=0;
266.
267.    %Variables para compensar desequilibrios
268.    Global_IR_ca=0;
269.    Global_IS_ca=0;
270.    Global_IT_ca=0;
271.
272.    %Variables para compensar reactiva
273.    Global_IT_ca_reactiva=0;
274.    Global_IR_ca_reactiva=0;
275.    Global_IS_ca_reactiva=0;
276.    Global_escalado_react=0;
277.    I_MAXIMA_NUEVA=I_MAXIMA_COMPENSA;
278.    FacUtil_F_R=0;
279.    FacUtil_F_S=0;
280.    FacUtil_F_T=0;
281.    FacUtil_Total=0;
282.
283.    % Rutina que se ejecuta segun tiempo de muestreo
284.    elseif flag==3
285.
286.        if t > (xnnx*xTTm)
287.            xnnx = xnnx+1;
288.
289.            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
290.            %%%%%% Lecturade Valores de Entrada....
291.            I_MAXIMA_COMPENSA = u(73);
292.            NUM_ALGORITMO = u(74);
293.
294.            %%%%%%%%%%%%%%%%%%%%%%%%
295.            % FASE TRIFASICA-R
296.            % Armonico-1-Fundamental
297.            IR_1_Mag = u(1); IR_1_Fase = u(1+11); %Se suma 11 por como
   está la
```

```
298.            %señal de fase en el bus con respecto a la magnitud del
   mismo armónico
299.            % Armonico-3º
300.            IR_3_Mag = u(2); IR_3_Fase = u(2+11);
301.            % Armonico-5º
302.            IR_5_Mag = u(3); IR_5_Fase = u(3+11);
303.            % Armonico-7º
304.            IR_7_Mag = u(4); IR_7_Fase = u(4+11);
305.            % Armonico-9º
306.            IR_9_Mag = u(5); IR_9_Fase = u(5+11);
307.            % Armonico-11º
308.            IR_11_Mag = u(6); IR_11_Fase = u(6+11);
309.            % Armonico-13º
310.            IR_13_Mag = u(7); IR_13_Fase = u(7+11);
311.            % Armonico-15º
312.            IR_15_Mag = u(8); IR_15_Fase = u(8+11);
313.            % Armonico-17º
314.            IR_17_Mag = u(9); IR_17_Fase = u(9+11);
315.            % Armonico-19º
316.            IR_19_Mag = u(10); IR_19_Fase = u(10+11);
317.            % Armonico-21º
318.            IR_21_Mag = u(11); IR_21_Fase = u(11+11);
319.
320.            %%%%%%%%%%%%%%%%%%%%%%
321.            % FASE TRIFASICA-S
322.            % Armonico-1-Fundamental
323.            IS_1_Mag = u(23); IS_1_Fase= u(23+11);
324.            % Armonico-3º
325.            IS_3_Mag = u(24); IS_3_Fase = u(24+11);
326.            % Armonico-5º
327.            IS_5_Mag= u(25); IS_5_Fase= u(25+11);
328.            % Armonico-7º
329.            IS_7_Mag= u(26); IS_7_Fase = u(26+11);
330.            % Armonico-9º
331.            IS_9_Mag = u(27); IS_9_Fase = u(27+11);
332.            % Armonico-11º
333.            IS_11_Mag = u(28); IS_11_Fase = u(28+11);
334.            % Armonico-13º
335.            IS_13_Mag = u(29); IS_13_Fase = u(29+11);
336.            % Armonico-15º
337.            IS_15_Mag= u(30); IS_15_Fase= u(30+11);
338.            % Armonico-17º
339.            IS_17_Mag= u(31); IS_17_Fase = u(31+11);
340.            % Armonico-19º
341.            IS_19_Mag = u(32); IS_19_Fase = u(32+11);
342.            % Armonico-21º
343.            IS_21_Mag = u(33); IS_21_Fase = u(33+11);
344.
345.            %%%%%%%%%%%%%%%%%%%%%%
346.            % FASE TRIFASICA-T
347.            % Armonico-1-Fundamental
348.            IT_1_Mag= u(45); IT_1_Fase= u(45+11);
349.            % Armonico-3º
350.            IT_3_Mag = u(46); IT_3_Fase = u(46+11);
351.            % Armonico-5º
352.            IT_5_Mag = u(47); IT_5_Fase = u(47+11);
```

```
353.            % Armonico-7º
354.            IT_7_Mag = u(48); IT_7_Fase = u(48+11);
355.            % Armonico-9º
356.            IT_9_Mag = u(49); IT_9_Fase = u(49+11);
357.            % Armonico-11º
358.            IT_11_Mag = u(50); IT_11_Fase = u(50+11);
359.             % Armonico-13º
360.            IT_13_Mag = u(51); IT_13_Fase = u(51+11);
361.            % Armonico-15º
362.            IT_15_Mag = u(52); IT_15_Fase = u(52+11);
363.            % Armonico-17º
364.            IT_17_Mag = u(53); IT_17_Fase = u(53+11);
365.            % Armonico-19º
366.            IT_19_Mag = u(54); IT_19_Fase = u(54+11);
367.            % Armonico-21º
368.            IT_21_Mag = u(55); IT_21_Fase = u(55+11);
369.            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
370.            % I RST
371.               %MAGNITUD-----------FASE
372.            % Secuencia-positiva-Fundamental
373.            Isp_Mag=u(67); Isp_Fase=u(68);
374.            % Secuencia-negativa-Fundamental
375.            Isn_Mag=u(69); Isn_Fase=u(70);
376.            % Secuencia-cero-Fundamental
377.            Is0_Mag=u(71); Is0_Fase=u(72);
378.
379.            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
380.            % Algoritmo de obtencion de las
381.            % Corrientes de Referncia de la 3 Fases
382.            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
383.
384.            I_R_Ref_COMPENSA=zeros(12,1);%Inicializa las corrientes de
   compensación
385.            I_S_Ref_COMPENSA=zeros(12,1);
386.            I_T_Ref_COMPENSA=zeros(12,1);
387.
388.            %se obtiene los valores de las corrientes de desequilibrio
389.            %Componentes de secuencia directa, inversa y homopolar
390.            IRd=Isp_Mag*exp(1j*(Isp_Fase*pi/180))
391.            IRi=Isn_Mag*exp(1j*(Isn_Fase*pi/180))
392.            IRh=Is0_Mag*exp(1j*(Is0_Fase*pi/180))
393.            as=(-1/2)+sqrt(3)/2*1j %a = 1<120 y a^2= 1<240
394.
395.            %Diseño del Compensador Activo
396.            %Calculo de corrientes de referencia del filtro activo para
397.            %%compensación de desequilibrios
398.            IR_ca=IRi + IRh;
399.            IS_ca=as*IRi + IRh;
400.            IT_ca=as^2*IRi + IRh;
401.
402.            %=====================REACTIVA==========================
403.            %Calculo de corrientes de referencia del filtro activo para
404.            %Compensación de Reactiva.
405.            IRdr=Isp_Mag*sin(Isp_Fase*pi/180)*j
406.
407.            IR_ca_reactiva=IRdr
```

```
408.            IS_ca_reactiva=IRdr*as^2
409.            IT_ca_reactiva=IRdr*as
410.
411.            I_REACT=Maximo3Fases([abs(IR_ca_reactiva)
   abs(IS_ca_reactiva) abs(IT_ca_reactiva)])
412.            %obtención de la amplitud máxima de entre las 3 fases de las
413.            %corrientes de reactiva.
414.
415.            %======================REACTIVA=========================
416.
417.            I_DESQ=Maximo3Fases([abs(IR_ca) abs(IS_ca) abs(IT_ca)])
   %obtención de la
418.            %amplitud máxima de entre las 3 fases de las corrientes de
   desequilibrios
419.
420.            I_RESTANTE=I_MAXIMA_COMPENSA%Se emplea en caso de que haya
   desequilibrios o no
421.
422.            factcmp=zeros([12,1])%zeros([length(M),1]);%Factor de
   compensación
423.            factcmp_reord=zeros([12,1])%zeros([length(M),1])%Factor de
   compensación
424.            %reordenado cuando se usa el algoritmo Qsort
425.
426.            if I_DESQ<I_MAXIMA_COMPENSA %Sirve para indicar si se pueden
   compensar
427.                %desequilibrios y en caso q haya como los compensa
   teniendo en cuenta
428.                %se tiene menos corriente para compensar armónicos
429.                if I_DESQ<=Umbral_Desq
430.
431.                    factcmp_reord(1,1)=0%Indica q no se estan
   compensando desequilibrios
432.                    %Compensar Desequilibrios
433.                    %se asigna el Valor de las corrientes de
   desequilibrio
434.                    I_R_Ref_COMPENSA(1,1)=0%
435.                    I_S_Ref_COMPENSA(1,1)=0%
436.                    I_T_Ref_COMPENSA(1,1)=0%
437.
438.                    Global_IR_ca=0;%Corrientes de referencia Globales
439.                    Global_IS_ca=0;%La salida no envia valores de
   compesacion
440.                    Global_IT_ca=0;
441.                    I_RESTANTE=I_MAXIMA_COMPENSA
442.
443.                else
444.                    factcmp_reord(1,1)=1%Indica q se estan compensando
   desequilibrios
445.                    %Compensar Desequilibrios
446.                    %se asigna el Valor de las corrientes de
   desequilibrio
447.                    I_R_Ref_COMPENSA(1,1)=abs(IR_ca)%
448.                    I_S_Ref_COMPENSA(1,1)=abs(IS_ca)%
449.                    I_T_Ref_COMPENSA(1,1)=abs(IT_ca)%
450.
```

```
451.                     Global_IR_ca=IR_ca;%Corrientes de referencia
   Globales
452.                     Global_IS_ca=IS_ca;
453.                     Global_IT_ca=IT_ca;
454.
455.                     I_RESTANTE=I_MAXIMA_COMPENSA-I_DESQ
456.                 end
457.             end
458.
459.                 ARM=[0.0 0.0 0.0; %Valores de la fundamental q entrega
   la red
460.                     IR_3_Mag  IS_3_Mag   IT_3_Mag;
461.                     IR_5_Mag  IS_5_Mag   IT_5_Mag;
462.                     IR_7_Mag  IS_7_Mag   IT_7_Mag;
463.                     IR_9_Mag  IS_9_Mag   IT_9_Mag;
464.                     IR_11_Mag IS_11_Mag  IT_11_Mag;
465.                     IR_13_Mag IS_13_Mag  IT_13_Mag;
466.                     IR_15_Mag IS_15_Mag  IT_15_Mag;
467.                     IR_17_Mag IS_17_Mag  IT_17_Mag;
468.                     IR_19_Mag IS_19_Mag  IT_19_Mag;
469.                     IR_21_Mag IS_21_Mag  IT_21_Mag;
470.                     ]
471.
472.                 ARM_Fase=[0.0 0.0 0.0; %Valores de fase q la fundamental
   q entrega la red
473.                     IR_3_Fase  IS_3_Fase   IT_3_Fase;
474.                     IR_5_Fase  IS_5_Fase   IT_5_Fase;
475.                     IR_7_Fase  IS_7_Fase   IT_7_Fase;
476.                     IR_9_Fase  IS_9_Fase   IT_9_Fase;
477.                     IR_11_Fase IS_11_Fase  IT_11_Fase;
478.                     IR_13_Fase IS_13_Fase  IT_13_Fase;
479.                     IR_15_Fase IS_15_Fase  IT_15_Fase;
480.                     IR_17_Fase IS_17_Fase  IT_17_Fase;
481.                     IR_19_Fase IS_19_Fase  IT_19_Fase;
482.                     IR_21_Fase IS_21_Fase  IT_21_Fase;
483.                     ]
484.
485.                 M = Maximo3Fases(ARM)%Obtiene el mayor valor de
   corriente de entre
486.                 %las 3 fases para cada armónico
487.
488.                 GlobalM=M;
489.                 M(1,1)=100000;%Para q el fundamental sea el mayor de
   todos siempre
490.                 for i=2:length(M)%Para los armónicos menores que el
   umbral hacerlos cero
491.                     if M(i)<Umbral
492.                         M(i)=0
493.                     end
494.                 end
495.
496.                 switch NUM_ALGORITMO %PARA ELEGIR CON Q ALGORITMO
   COMPENSAR
497.                     case 1
498.                         fprintf('Se esta ejecutando el algoritmo 1 \n')
499.
```

```
                        for i=2:length(M)%for para saber hasta que armonico compensar,
                            %siendo el fundamental el mayor siempre se empieza desde la ubicacion 2

                            I_ref=I_ref+M(i)%se suma los valores de los armónicos

                            if I_ref>I_RESTANTE %Cuando las amplitudes de los armonicos
                                %sobrepasan la corriente máxima de compensación se aplica
                                %un factor de escalado
                                factcmp(i)=1-((abs(I_RESTANTE-I_ref)/M(i,1)));%Se obtiene el valor
                                %del factor de compensación q necesita ser escalado
                                I_ref=sum(M([2:i-1],1))+M(i).*factcmp(i);%El total de la
                                %corriente que se va a compensar teniendo en cuenta el ultimo
                                %armonico que ha sido escalado
                                break
                            end

                            if M(i)==0 %para que el factor de compensación sea cero
                                %cuando el armónico sea cero o este debajo del umbral
                                factcmp(i)=0;
                            else
                                factcmp(i)=1;
                            end
                        end

                        factcmp_reord([2:11],1)=factcmp([2:11],1)%Se envía el factor de compensación

                    case 2
                        fprintf('Se esta ejecutando el algoritmo 2 \n');

                        %Algoritmo Qsort para ordenar de mayor
                        %a menor el valor de los armónicos
                        [ML,I]=MiQsortDescendente(M)

                        for i=2:length(I)%for para saber hasta que armonico compensar,
                            %siendo el fundamental el mayor siempre se empieza desde la ubicacion 2

                            I_ref=I_ref+ML(i)%se suma los valores de los armónicos

                            if I_ref>I_RESTANTE %Cuando las amplitudes de los armonicos sobrepasan
```

```
539.                                %la corriente máxima de compensación se
   aplica un factor de escalado
540.                                factcmp(i)=1-((abs(I_RESTANTE-
   I_ref)/ML(i,1)));%Se obtiene el valor
541.                                %del factor de compensación q necesita
   ser escalado
542.                                I_ref=sum(ML([2:i-
   1],1))+ML(i).*factcmp(i)%El total de la corriente
543.                                %que se va a compensar teniendo en
   cuenta el ultimo armonico que ha
544.                                %sido escalado
545.                                break
546.                            end
547.
548.                            if ML(i)==0 %para que el factor de
   compensación sea cero cuando el armónico
549.                                %sea cero o este debajo del umbral
550.                                factcmp(i)=0;
551.                            else
552.                                factcmp(i)=1;
553.                            end
554.                        end%fin del for
555.
556.                        for i=2:length(I)% Vextor de factores de
   compensaciin reordenada,
557.                            %no se toma en cuenta la fundamental
558.                            factcmp_reord(I(i),1)=factcmp(i);
559.                        end
560.
561.                    otherwise
562.                        for i=1:length(GlobalM)% Vextor de factores de
   compensacion reordenada,
563.                            %no se toma en cuenta la fundamental
564.                            factcmp_reord(i,1)=0;
565.                        end
566.                        Global_IR_ca=0;%Corrientes de referencia
   Globales
567.                        Global_IS_ca=0;
568.                        Global_IT_ca=0;
569.
570.                        I_R_Ref_COMPENSA(1,1)=0;
571.                        I_S_Ref_COMPENSA(1,1)=0;
572.                        I_T_Ref_COMPENSA(1,1)=0;
573.
574.                end %Fin del Switch
575.
576.                %Obtengo la Corriente sobrante si es q la hay pa
   compensar
577.                %reactiva
578.                I_RESTANTE_REACTIVA=I_RESTANTE-I_ref%Obtiene la
   corriente
579.                %que sobra para compensar reactiva.
580.
581.
   %%%%%%%%%%%%%%%%%%%%%%%%REACTIVA%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
582.                if I_REACT~=0
```

```
583.                     if I_RESTANTE_REACTIVA>0
584.                         %Compensar Desequilibrios
585.                         %se asigna el Valor de las corrientes de
   desequilibrio
586.                         escalado_react=I_RESTANTE_REACTIVA/I_REACT
587.
588.                         if escalado_react<1
589.                             factcmp_reord(12,1)=escalado_react
590.                             %factcmp_reord(12,1) es el escalado que va a
   la
591.                             %tabla posicion (12,1)
592.                         else
593.                             factcmp_reord(12,1)=1
594.                         end
595.
596.                         Global_escalado_react=factcmp_reord(12,1)
597.
598.                         %Corrientes de reactiva que van a la tabla
   posicion (12,1)
599.
   I_R_Ref_COMPENSA(12,1)=abs(IR_ca_reactiva)*factcmp_reord(12,1)%
600.
   I_S_Ref_COMPENSA(12,1)=abs(IS_ca_reactiva)*factcmp_reord(12,1)%
601.
   I_T_Ref_COMPENSA(12,1)=abs(IT_ca_reactiva)*factcmp_reord(12,1)%
602.
603.                         Global_IR_ca_reactiva=IR_ca_reactiva;%Corrientes
   de referencia Globales
604.                         Global_IS_ca_reactiva=IS_ca_reactiva;
605.                         Global_IT_ca_reactiva=IT_ca_reactiva;
606.
607.                         %factcmp_reord(12,1)=escalado_react%Indica q se
   estan compensando desequilibrios
608.
609.                         I_RESTANTE=I_MAXIMA_COMPENSA-I_DESQ
610.                     end
611.                 end
612.
   %%%%%%%%%%%%%%%%%%%%%%REACTIVA%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
613.                 I_ref=0;%inicializo la corriente de referencia
614.
615.                 %GobalI=I; %Global de la matriz de indices
616.                 GlobalPh=ARM_Fase; %Global de la matriz de Fase de los
   Armónicos
617.
618.                 for i=2:11 %para asignar el Valor de las Corrientes de
   compensación de armónicos
619.                     I_R_Ref_COMPENSA(i,1)=ARM(i,1).*factcmp_reord(i);
620.                     I_S_Ref_COMPENSA(i,1)=ARM(i,2).*factcmp_reord(i);
621.                     I_T_Ref_COMPENSA(i,1)=ARM(i,3).*factcmp_reord(i);
622.                 end
623.
624.
   FacUtil_F_R=sum(I_R_Ref_COMPENSA(1:12,1))/I_MAXIMA_COMPENSA;%Factor de
   Utilizacion de la Fase R
```

```
625.
   FacUtil_F_S=sum(I_S_Ref_COMPENSA(1:12,1))/I_MAXIMA_COMPENSA;%Factor de
   Utilizacion de la Fase S
626.
   FacUtil_F_T=sum(I_T_Ref_COMPENSA(1:12,1))/I_MAXIMA_COMPENSA;%Factor de
   Utilizacion de la Fase T
627.
628.
   FacUtil_Total=(FacUtil_F_R+FacUtil_F_S+FacUtil_F_T)/3%Factor de
   Utilización Total
629.
630.                primeravez=1;
631.                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
632.        end
633.
634.        if(t>tmodulador*(1/400000))
635.            tmodulador=tmodulador+1
636.
637.            if(primeravez==1)
638.                f=50;
639.                w=2*pi*f;
640.                CORR_fase=0.1;
641.
642.                for i=1:length(GlobalM)
643.                    a(i,1)= I_R_Ref_COMPENSA(i,1)*sin((2*i-
   1)*w*t+((GlobalPh(i,1).*pi/180)+((CORR_fase*i)*pi/180)));
644.                    %se toma el cuenta el desplazamiento de la
645.                    %fundamental como factor de correcion de fase con
   respecto de fase 0
646.                    b(i,1)= I_S_Ref_COMPENSA(i,1)*sin((2*i-
   1)*w*t+((GlobalPh(i,2).*pi/180)+((CORR_fase*i)*pi/180)));
647.                    %se toma el cuenta el desplazamiento de la
648.                    %fundamental como factor de correcion de fase se
   suma 120 para obtener
649.                    %el desplazamiento
650.                    c(i,1)= I_T_Ref_COMPENSA(i,1)*sin((2*i-
   1)*w*t+((GlobalPh(i,3).*pi/180)+((CORR_fase*i)*pi/180)));
651.                    %se toma el cuenta el desplazamiento de la
652.                    %fundamental como factor de correcion de fase se
   resta 120 para obtener
653.                    %el desplazamiento
654.                end
655.
656.                %corrientes de referencia para desequilibrios a
   compensar
657.
   DES_R=(abs(Global_IR_ca)*sin(w*t+angle(Global_IR_ca)))*factcmp_reord(1,1
   );%
658.
   DES_S=(abs(Global_IS_ca)*sin(w*t+angle(Global_IS_ca)))*factcmp_reord(1,1
   );%
659.
   DES_T=(abs(Global_IT_ca)*sin(w*t+angle(Global_IT_ca)))*factcmp_reord(1,1
   );%
660.
661.                %corrientes de referencia para reactiva a compensar
```

```
662.
   REACT_R=(abs(Global_IR_ca_reactiva)*Global_escalado_react)*sin(w*t+angle
   (Global_IR_ca_reactiva));%
663.
   REACT_S=(abs(Global_IS_ca_reactiva)*Global_escalado_react)*sin(w*t+angle
   (Global_IS_ca_reactiva));%
664.
   REACT_T=(abs(Global_IT_ca_reactiva)*Global_escalado_react)*sin(w*t+angle
   (Global_IT_ca_reactiva));%
665.
666.                senal(1,1)=sum(a([2:11],1));
667.                senal(2,1)=sum(b([2:11],1));
668.                senal(3,1)=sum(c([2:11],1));
669.                senal(4,1)=DES_R;
670.                senal(5,1)=DES_S;
671.                senal(6,1)=DES_T;
672.                senal(7,1)=REACT_R;
673.                senal(8,1)=REACT_S;
674.                senal(9,1)=REACT_T;
675.                senal(10,1)=FacUtil_F_R;
676.                senal(11,1)=FacUtil_F_S;
677.                senal(12,1)=FacUtil_F_T;
678.                %senal(10,1)=FacUtil_Total;
679.            end %End if primera vez
680.        end % End if modulador
681.        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
682.        % SALIDA del SISTEMA
683.        sys
   =[I_R_Ref_COMPENSA;I_S_Ref_COMPENSA;I_T_Ref_COMPENSA;factcmp_reord;senal
   ];
684.
685.    elseif flag==4
686.    sys = [];
687.    else
688.    sys = [];
689.    end
690.    % end mdlTerminate
```

ANEXO III

3. CODIGO IMPLEMENTADO EN EL MICROCONTROLADOR

CÓDIGO MAIN

```
#include "F2806x_Device.h"      // Fichero include: F2806x Headerfile
#include "PeripheralHeaderIncludes.h"
#include "F2806x_Examples.h"    // Fichero include: F2806x Examples

#include "Settings_APLI.h"
#include "Union_datos.h"
#include "Serial_HW.h"
#include "ModbusMaster.h" /*Servicio modbusmaster  por 485 half duplex*/
#include "ModbusSlave.h" /*Servicio modbus esclavo, este por usb*/
#include "ALGORITMO_COMUNICACION_MODBUS.h"
#include "ALGORITMO_OPTIMIZADOR.h"

//#include "Timer_obj.h"    // Declaracion del OBJETO: Timer
#include "GLOBAL_DEFS.h"  // Declaracion de definiciones utilizadas en el programa
#include "GLOBAL_VARS.h"  // Declaracion de variables globales del programa
#include "GLOBAL_FUNCS.h" // Declaracion de Funciones prototipo del programa
#include "Funcs_APLI_Modbus.h" // Funciones y variables utilizadas en esta
aplicacion y relacionadas con el MODBUS

// Funciones prototipos declaradas en este fichero relacionadas con el Hardware-
2806x
#include "FUNCS_PROTIPO_HW.h"
#include "COMANDOS.h"
////////////////////////////////////////////////////////////////////////////
/// DECLARACION DE VARIABLES GLOBALES .....
int Backticker=0;
unsigned int todo_ok =0;

//////////////////////////////////
//  PROGRAMA PRINCIPAL
////////////////////////////////////////////////////////////////////////////
void main(void){
```

```
	//////////////////////////////////////////////////////////////////
	//     INICIALIZACIÓN DEL SISTEMA
	Configurar_Hardware_DSC();

	// Inicializar variables de la Aplicacion...
	Inicializar_VAR_Aplicacion();

	//////////////////////////////////////////////////////////////////
	todo_ok =1; // Variable sincronizacion de arranque de los Timers..

	//////////////////////////////////////
	//     BUCLE SIN FIN .....
	while(1) {

		Backticker++;

		// Testeo (polling de datos recibidos en SCI-A) ..............
		// mediante el protocolo-MODBUS MODO ESCLAVO. NO Utilizado ????
		mb.loopStates(&mb);

		// Testeo (polling de datos recibidos en SCI-B)
		// mediante el protocolo-MODBUS MODO MAESTRO, COMUNICACION FILTRO
		mb_master.loopStates(&mb_master);



		// Hacer MEF del Algortimo de COMUNCIACIONES READ-WRITE.....

	Algoritmo_COMUNICACION_MODBUS.MEF_ALGORIT(&Algoritmo_COMUNICACION_MODBUS);

		// Hacer MEF del Algortimo de OPTIMIZACION de COMPENSACION.....
		//Algoritmo_OPTI.MEF_ALGORIT(&Algoritmo_OPTI);

	} // FIN del BUCLE SIN FIN

} // Fin del MAIN
////////////////////////////////////////////////
/     INTERRUPCIÓN TIMER 2 CADA 100us
interrupt void TC0_Cpu_Timer2_isr(void) {

	VTimer2++;

	if((todo_ok)%1==1){  // TICK del MODBUS_MASTER (100us)
			mb.timer.Tick(&mb.timer);
	}

	if((todo_ok)%100==1){ // TICK del OBJETO_OPTIMIZADOR (10ms)
			Algoritmo_OPTI.Timer_OPTI.Tick(&Algoritmo_OPTI.Timer_OPTI);
	}
	//////////////////////////////////////////
	CpuTimer2Regs.TCR.bit.TIF = 1;  // clear flag
}
//////////////////////////////////////////////////////////////////////
void Inicializar_VAR_Aplicacion(void){
```

```
	// Inicializar el constructor del MODBUS esclavo... Para cnectarse con el
MONITOR-PC
	construct_ModbusSlave_con_callback(&mb,CANAL_SCI_A,57600,SERIAL_PARITY_NONE,
COMUNICACION_Parser_MODBUS,&ID_ESCLAVO,0);

	// Constructor del MODBUS Master...
	construct_ModbusMaster(&mb_master,CANAL_SCI_B,57600,SERIAL_PARITY_NONE,&FALL
O_MODBUS,0,callback_modbus);


	// Constructor del ALGORITMO DE COMUNICACION READ-WRITE
	Construct_Algoritmo_COMUNI(&Algoritmo_COMUNICACION_MODBUS,&mb_master);

	// Constructor del ALGORITMO EXPERTO DE COMPENSACION...
	Construct_Algoritmo_COMPENSA (&Algoritmo_OPTI,
&Algoritmo_COMUNICACION_MODBUS);

	// Tiempo de ESpera entre COMPENSACION en al Algoritmo de COMPENSACION...
	Algoritmo_OPTI.Timer_OPTI.setTimerReloadPeriod(&Algoritmo_OPTI.Timer_OPTI,30
0); // Recargar con 300 Ticks = 3000ms


		////////////////////////////////

}

//////////////////////////////////////////////////////////////////////////
// FIN DEL CODIGO DEL FICHERO .....
//////////////////////////////////////////////////////////////////////////
```

CÓDIGO ALGORITMO OPTIMIZADOR.C

```
#include "ALGORITMO_OPTIMIZADOR.h"
#include "math.h"       /* floor */
#include "STR_Compartida_Filtro_Activo.h"

// Status del comando "Statte" cuya dir:200
#define Stat_FA_Fallo           0
#define Stat_FA_Operacion   2
#define Stat_FA_OFF             3
#define Stat_FA_Full_Load       4

////////////////////////////////////////////////////////////////////////////////
/
// Definición de funciones PROTIPOS en est Fichero: ALGORITMO_OPTIMIZADOR.c
void maximo(float ARM[][3]);//Funcion maximo de 3 fases
void ordenamiento(float A[], int I[], int primero, int ultimo);//Funcion de
ordenamiento ascendente sin indices
void qsort_Arm(float A[]);  // Algortimo de ordenacion de Mayor --> Menor de los
Armonicos ....

static float max(float a, float b, float c);
////////////////////////////////////////////////////////////////////////
// Declaracion de las funciones prototipos de la MEF en este fichero:
static void Esta_Direct_ON_Filtro (STR_ALGORITMO_OPTI *Obj_OPTI);
static int Alg_Compensa_DESEQUILIBRIOS(STR_ALGORITMO_OPTI *self);
static int Alg_Compensa_ARMONICOS(STR_ALGORITMO_OPTI *self);
static int Alg_Compensa_REACTIVA(STR_ALGORITMO_OPTI *self);

////////////////////////////////////////////////////////////////////////////
// ALGORITMOS DE LA MAQUINAS DE ESTADOS DEL OPTIMIZADOR
////////////////////////////////////////////////////////////////////////////
static void Esta_Direct_ON_Filtro (STR_ALGORITMO_OPTI *Obj_OPTI){

      // Comprobamos que el filtro esta a 1 DIRECT_ON ...
      if(ESTRUCTURA_Compartida_FA.Lecturas.Direct_ON_OFF_u16 == 1) {
                  Obj_OPTI->Flg_MEF.CONEXION_FA = 1;

      } else Obj_OPTI->Flg_MEF.CONEXION_FA = 0;

}

////////////////////////////////////////////////////////////////////////
static int Alg_Compensa_DESEQUILIBRIOS(STR_ALGORITMO_OPTI *Obj_OPTI){
      int Status_FA=1;

      ////////////////////////////////////////////////////////////////////////
      // Leer el Valor de "Full_Load" de la Zona de datos Compartida entre Objeto
OPTI-COMUNI ...
      Status_FA = ESTRUCTURA_Compartida_FA.Lecturas.Estado_u16;

      // Es Full_Load ???
      Obj_OPTI->Flg_MEF.Full_Load = (Status_FA == Stat_FA_Full_Load) ? 1:0;
```

```
	if ((Obj_OPTI->Flg_MEF.Full_Load == 1)){ // Es Full_Load ???
			// DesActivar la Compensación de los Armonicos y Reactiva ....

	ESTRUCTURA_Compartida_FA.Escrituras.Compensacion_Armonica_ON_OFF_u16 = 0;
			ESTRUCTURA_Compartida_FA.Escrituras.P_Cien_Reactiva_f32 = 0.0;

			// Escribir en Datos compartidos para activar la MEF de
comunicacion-Modbus
			Obj_OPTI->_Obj_COMUNI->Flg_MEF.ACCI_ESCRIBIR=1;

			Obj_OPTI->_Estado = PROCESO_COMPENSA_CONEXION_FILTRO;

	}else {  // Sino es Full_Load ==> Hay corriente para compensar ...
			Obj_OPTI->Flg_MEF.Full_Load = 0;
			Obj_OPTI->_Estado = PROCESO_COMPENSA_ARMONICOS; // Prioridad
para COMPENSAR
		}
 return(0);
}


////////////////////////////////////////////////////////////////////////
static int Alg_Compensa_ARMONICOS(STR_ALGORITMO_OPTI *Obj_OPTI){
	//float Val_Aux;
	return(0);

}

////////////////////////////////////////////////////////////////////////
static int Alg_Compensa_REACTIVA(STR_ALGORITMO_OPTI *Obj_OPTI){
	int Status_FA=1;

	//////////////////////////////////////////////////////////////////////
	// Leer el Valor de "Full_Load" de la Zona de datos Compartida entre Objeto
OPTI-COMUNI ...
	Status_FA = ESTRUCTURA_Compartida_FA.Lecturas.Estado_u16;

	// Es Full_Load ???
	Obj_OPTI->Flg_MEF.Full_Load = (Status_FA == Stat_FA_Full_Load) ? 1:0;

	//////////////////////////////////////////////////////////////////////
	if (Obj_OPTI->Flg_MEF.Full_Load == 1){
		// Decrementar Porcentaje del factor de compensacion de reactiva ...
		ESTRUCTURA_Compartida_FA.Lecturas.P_Cien_Reactiva_u32 = \

	ESTRUCTURA_Compartida_FA.Lecturas.P_Cien_Reactiva_u32 - 5;
		return(2);

	}else {		// Sino HAy Full Load y % de la compensacion menor del 90% ==>
Incrementar
		if (ESTRUCTURA_Compartida_FA.Lecturas.P_Cien_Reactiva_u32 < 90){
			//Incrementar Porcentaje del factor de compensacion de
reactiva ...
			ESTRUCTURA_Compartida_FA.Lecturas.P_Cien_Reactiva_u32 = \
```

```
      ESTRUCTURA_Compartida_FA.Lecturas.P_Cien_Reactiva_u32 + 5;
                }

            }
      ////////////////////////////////////////////////////////////////////////////
////////////////////////////
      return(0);
}

//////////////////////////////////////////////////////////
// MEF de la Algoritmo de OPTIMIZACION.....
//////////////////////////////////////////////////////////
static void Algoritmo_MEF_OPTI(STR_ALGORITMO_OPTI *Obj_OPTI){
      int Status_FA=1;
      int dev_rut=0;
      float Val_Aux;

                  switch(Obj_OPTI->_Estado){

                              // En este ESTADO se realizan SI esta CONECTADO el
FILTRO-ACTIVO
                              case PROCESO_COMPENSA_CONEXION_FILTRO:

                                    Obj_OPTI->_Operacion_Compensa =
OP_CONEXION_FILTRO;

                                  Esta_Direct_ON_Filtro (Obj_OPTI);

                                    if (Obj_OPTI->Flg_MEF.CONEXION_FA == 1){
                                          // Escribir para la Compensación de
Desequilibrios ...

      ESTRUCTURA_Compartida_FA.Escrituras.Compensacion_Desequilibrios_ON_OFF_u16 =
1;
                                          // Escribir en Datos compartidos para
activar la MEF de comunicacion-Modbus
                                          Obj_OPTI->_Obj_COMUNI-
>Flg_MEF.ACCI_ESCRIBIR=1;

                                          Obj_OPTI->_Estado =
PROCESO_COMPENSA_DESEQUILIBRIOS;
                                    }

                                    break;


      ////////////////////////////////////////////////////////////////////////////
////////////
                              // En este ESTADO se realizan la COMPENSACION de
los DESEQUILIBRIOS en el FILTRO-ACTIVO
                              // y escribe (MODBUS)las consignas de compensacion
en el filtro
                              case PROCESO_COMPENSA_DESEQUILIBRIOS:
```

```
                                    Obj_OPTI->_Operacion_Compensa =
OP_CONEXION_FILTRO ;

                                    // Activer TIMER ...
                                    Obj_OPTI->Timer_OPTI.start(&Obj_OPTI-
>Timer_OPTI);


      ////////////////////////////////////////////////////////////////////////
                                    // Esperar 30 segundos ....
                                    if (!(Obj_OPTI-
>Timer_OPTI.expiradoTimer(&Obj_OPTI->Timer_OPTI)))return;

                                    // PArar y Resetear el TIMER...
                                    Obj_OPTI->Timer_OPTI.stop(&Obj_OPTI-
>Timer_OPTI);
                                    Obj_OPTI->Timer_OPTI.resetTimer(&Obj_OPTI-
>Timer_OPTI);


      ////////////////////////////////////////////////////////////////////////
                                    dev_rut =
Alg_Compensa_DESEQUILIBRIOS(Obj_OPTI);

                                    return;

                                // En este ESTADO se realizan la COMPENSACION de
los ARMONICOS en el FILTRO-ACTIVO
                                // y escribe (MODBUS)las consignas de compensacion
en el filtro
                                case PROCESO_COMPENSA_ARMONICOS:

                                    Obj_OPTI->_Operacion_Compensa =
OP_COMPENSA_DESQUILIBRIOS ;

                                    // Comprobar el parametro: THDi, si mayor
del 5% Activar compensacion ARMONICOS
                                    Val_Aux =
ESTRUCTURA_Compartida_FA.Lecturas.THDi_f32;

                                    // Comparar THDi con el Máximo del Filtro
Activo ==> 5% ...
                                    if (Val_Aux > 5.0 ) { // Si es posible
Compensar ARMONICOS ...
                                          dev_rut =
Alg_Compensa_ARMONICOS(Obj_OPTI);
                                                // Activer TIMER ...
                                                Obj_OPTI-
>Timer_OPTI.start(&Obj_OPTI->Timer_OPTI);

                                    } else { // Intentar Compensar REACTIVA
                                                // Escribir para Activar la
Compensación de REACTIVA ...

      ESTRUCTURA_Compartida_FA.Escrituras.Cos_fi_Reactiva_f32 = 1;
```

```
                                                    // Escribir en Datos
compartidos para Activar la MEF de comunicacion-Modbus
                                                    Obj_OPTI->_Obj_COMUNI-
>Flg_MEF.ACCI_ESCRIBIR = 1;
                                                    }


      ////////////////////////////////////////////////////////////////////////
                                        // Esperar 30 segundos ....
                                        if (!(Obj_OPTI-
>Timer_OPTI.expiradoTimer(&Obj_OPTI->Timer_OPTI)))return;

                                        // PArar y Resetear el TIMER...
                                        Obj_OPTI->Timer_OPTI.stop(&Obj_OPTI-
>Timer_OPTI);
                                        Obj_OPTI->Timer_OPTI.resetTimer(&Obj_OPTI-
>Timer_OPTI);


      ////////////////////////////////////////////////////////////////////////
                                        // Leer el Valor de "Full_Load" de la Zona
de datos Compartida entre Objeto OPTI-COMUNI ...
                                        Status_FA =
ESTRUCTURA_Compartida_FA.Lecturas.Estado_u16;
                                        Obj_OPTI->Flg_MEF.Full_Load = 0;

                                        if (Status_FA == Stat_FA_Full_Load){ // Es
Full_Load ???
                                                    Obj_OPTI->Flg_MEF.Full_Load =
1;
                                                    // DesActivar la Compensación
de los Armonicos ....

                                                    Obj_OPTI->_Estado =
PROCESO_COMPENSA_CONEXION_FILTRO;

                                        }else if (Obj_OPTI->Flg_MEF.Arm_Mayor_50
==1){ // Existe Potencia para compensar mas ????
                                                          Obj_OPTI->_Estado =
PROCESO_COMPENSA_REACTIVA;
                                                    }// else {incrementar el
siguiente armonico a compensar ....}


                                        break;

                                  // En este ESTADO se realizan la COMPENSACION de
la REACTIVA en el FILTRO-ACTIVO
                                  // y escribe (MODBUS)las consignas de compensacion
en el filtro
                                  case PROCESO_COMPENSA_REACTIVA:

                                        // Comprobar el parametro: PF, si menor de
0.95 Activar compensacion REACTIVA ...
```

```
                                        if
((ESTRUCTURA_Compartida_FA.Lecturas.PF_f32) > 0.95 ) {
                                              Obj_OPTI->_Estado =
PROCESO_COMPENSA_CONEXION_FILTRO;
                                              // PArar y Resetear el TIMER...
                                              Obj_OPTI->Timer_OPTI.stop(&Obj_OPTI-
>Timer_OPTI);
                                              Obj_OPTI-
>Timer_OPTI.resetTimer(&Obj_OPTI->Timer_OPTI);
                                              return;

                                        }else { // Compensar REACTIVA ...
                                                    Obj_OPTI->_Operacion_Compensa =
OP_COMPENSA_REACTIVA ;

                                                    dev_rut =
Alg_Compensa_REACTIVA(Obj_OPTI);

                                                    Obj_OPTI-
>Timer_OPTI.start(&Obj_OPTI->Timer_OPTI);


      ////////////////////////////////////////////////////////////////////////
                                                    // Esperar 30 segundos ....
                                                    if (!(Obj_OPTI-
>Timer_OPTI.expiradoTimer(&Obj_OPTI->Timer_OPTI)))return;

                                                    // PArar y Resetear el TIMER...
                                                    Obj_OPTI-
>Timer_OPTI.stop(&Obj_OPTI->Timer_OPTI);
                                                    Obj_OPTI-
>Timer_OPTI.resetTimer(&Obj_OPTI->Timer_OPTI);
                                              }

      ////////////////////////////////////////////////////////////////////////
////////
                                        if (dev_rut == 2){
                                              Obj_OPTI->_Estado =
PROCESO_COMPENSA_CONEXION_FILTRO;
                                              // PArar y Resetear el TIMER...
                                              Obj_OPTI->Timer_OPTI.stop(&Obj_OPTI-
>Timer_OPTI);
                                              Obj_OPTI-
>Timer_OPTI.resetTimer(&Obj_OPTI->Timer_OPTI);
                                        }
                                        return;

                                  ////////////////////////////////
                                  // ESTADO por Defecto .....
                                  default:
                                        break;
                        }


            }
```

```
////////////////////////////////////////////////////////////////////////
// Inicializacion del OBJETO ALGORITMO de Optimizacion ...
void Construct_Algoritmo_COMPENSA (STR_ALGORITMO_OPTI *Obj_OPTI,
STR_ALGORITMO_COMUNI *Obj_COMU){

	Obj_OPTI->_Estado = PROCESO_COMPENSA_CONEXION_FILTRO; //empezamos leyendo

	Obj_OPTI->_Obj_COMUNI = Obj_COMU;

	Obj_OPTI->MEF_ALGORITMO_OPTI = Algoritmo_MEF_OPTI;

	Construct_Timer_obj(&Obj_OPTI->Timer_OPTI);   //para controlar el tiempo de
espera entre la puesta en marcha de cada tipo de Compensacion

	Obj_OPTI->Estado_Algoritmo_ini = 1; //indicamos que acabamos de crear
}

////////////////////////////////////////////
// FIN del Codigo del Fichero !!!
////////////////////////////////////////////
```

CÓDIGO ALGORITMO OPTIMIZADOR.H

```
#ifndef INCLUDE_ALGORITMO_OPTIMIZADOR_H_
#define INCLUDE_ALGORITMO_OPTIMIZADOR_H_

////////////////////////////////////////////////////////////////////////////
#include "Timer_obj.h"
#include "ALGORITMO_COMUNICACION_MODBUS.h"

#define ULTI_ARMONICO_OPTI   49  // Ultimo armonico a OPTIMIZAR ...

typedef struct STR_ALGORITMO_OPTI STR_ALGORITMO_OPTI;

typedef enum { // ESTADOS de la MEF del Algoritmo COMPENSACION....
      PROCESO_COMPENSA_CONEXION_FILTRO,
      PROCESO_COMPENSA_DESEQUILIBRIOS,
      PROCESO_COMPENSA_ARMONICOS,
      PROCESO_COMPENSA_REACTIVA,
}STR_ALGORITMO_OPTI_ESTADO;


// Dentro del ESTADO-Lecturas indica si esta en proceso de Lectura o se han
finalizado...
typedef enum {
      OP_CONEXION_FILTRO,
      OP_COMPENSA_DESQUILIBRIOS,
      OP_COMPENSA_ARMONICOS,
      OP_COMPENSA_REACTIVA,

}OPERACION_ESTADO_COMPENSADOR;
////////////////////////////////////////////////////////////////
typedef struct {
                  unsigned CONEXION_FA:1;
                  unsigned Full_Load:1;
                  unsigned Arm_Mayor_50:1;
                  unsigned bit3:1;
                  unsigned bit4:1;
                  unsigned bit5:1;
                  unsigned bit6:1;
                  unsigned bit7:1;
                  unsigned bit8:1;
                  unsigned bit9:1;
                  unsigned bit10:1;
                  unsigned bit11:1;
                  unsigned bit12:1;
                  unsigned bit13:1;
                  unsigned bit14:1;
                  unsigned bit15:1;
} FLAGS_COMPENSADOR;
////////////////////////////////////////////////////////////////////////////
////////////
// Estructura del ALGORITMO DE OPTIMIZACION DE POTENCIA DEL CONVERTIDOR
////////////////////////////////////////////////////////////////////////////
////////////
```

```
struct STR_ALGORITMO_OPTI {

	STR_ALGORITMO_OPTI_ESTADO _Estado;

	OPERACION_ESTADO_COMPENSADOR _Operacion_Compensa;

	FLAGS_COMPENSADOR Flg_MEF;

	void(*MEF_ALGORITMO_OPTI)(STR_ALGORITMO_OPTI *);  // Puntero a la funcion de
ejecucion de la MEF del Propio Algoritmo...

	STR_ALGORITMO_COMUNI *_Obj_COMUNI; //puntero al Objeto de tratamiento de
Comunciaciones MODBUS ...

	Timer Timer_OPTI;   //para controlar el tiempo de espera entre la puesta en
marcha de cada tipo de Compensacion

	unsigned int Estado_Algoritmo_ini;  // Estado que indica que ha pasado al
menos una vez por el Algortimo...
};

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Constructor: Inicialziar la Estructura del Algortimo....
void Construct_Algoritmo_COMPENSA(STR_ALGORITMO_OPTI *self, STR_ALGORITMO_COMUNI
*Alg_COM);


#endif /* INCLUDE_APLI_ALGORITMO_H_ */
```

CÓDIGO ALGORITMO COMUNICACION_MODBUS.C

```
#include "ALGORITMO_COMUNICACION_MODBUS.h"
#include "math.h"       /* floor */
#include "STR_Compartida_Filtro_Activo.h"
#include "Union_datos.h"

////////////////////////////////////////////////////////////////////////////
// Declaración de variables globales ....
////////////////////////////////////////////////////////////////////////////

// Declaracion de Funciones Prototipos en este fichero ....
static void Leer_HR_F32(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES direccion);
static void Leer_HR_U16(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES direccion);
static void Test_Conexion_Canal_MODBUS_OK(STR_ALGORITMO_COMUNI *self,int
ID_Esclavo);
static void Escribir_HR_F32(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES direccion,
unsigned int *valor);
static void Escribir_HR_U16(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES direccion,
unsigned int valor);

static int Proceso_Ctrl_Lectura_Modbus_HR_U16(STR_ALGORITMO_COMUNI *Obj_COMUNI,
FA_DIRECCIONES direccion,unsigned int *);
static int Proceso_Ctrl_Lectura_Modbus_HR_F32(STR_ALGORITMO_COMUNI *Obj_COMUNI,
FA_DIRECCIONES direccion,float *);
static int Proceso_Ctrl_Escritura_Modbus_HR_U16(STR_ALGORITMO_COMUNI *Obj_COMUNI,
FA_DIRECCIONES direccion, unsigned int *Escribir_Dato_U16);
static int Proceso_Ctrl_Escritura_Modbus_HR_F32(STR_ALGORITMO_COMUNI *Obj_COMUNI,
FA_DIRECCIONES direccion, float *Escribir_Dato_U16);

static int Leer_Parametros_Simples(STR_ALGORITMO_COMUNI *Obj_COMUNI);
static int Leer_Parametros_Arrays(STR_ALGORITMO_COMUNI *Obj_COMUNI);
static int Leer_FFT_Arrays(STR_ALGORITMO_COMUNI *Obj_COMUNI, FA_DIRECCIONES
direccion,float *);
////////////////////////////////////////////////////////////////////////////
//array ordenado  de las direcciones en el MODBUS que realizan la gestion del
analisis armonico...
 unsigned int ARRAY_DIRECCIONES[] ={
FA_SEL_FFT_FASE,FA_ARM1,FA_ARM2,FA_ARM3,FA_ARM4,FA_ARM5,FA_ARM6,FA_ARM7,FA_ARM8,FA
_ARM9,\

      FA_ARM10,FA_ARM11,FA_ARM12,FA_ARM13,FA_ARM14,FA_ARM15,FA_ARM16,FA_ARM17,FA_A
RM18,FA_ARM19,FA_ARM20,FA_ARM21};

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
// Array de las direcciones donde se escribe en el Modbus el valor del % de la
Amplitud del armonico...
int ARRAY_ESCALADOS[] = {
COMP_ARM_ACTIVAR,COMP_H3,COMP_H5,COMP_H7,COMP_H9,COMP_H11,COMP_H13,COMP_H15,COMP_H
17,COMP_H19,COMP_H21,COMP_H23,COMP_H25,COMP_H27,COMP_H29,COMP_H31,COMP_H33,COMP_H3
5,COMP_H37,COMP_H39,COMP_H41,COMP_H43,COMP_H45,COMP_H47,COMP_H49 };
```

```
////////////////////////////////////////////////////////////////////////////////
//////////////////////777
 /* Funcion PRIMITIVA que genera la petición en modbus máster para leer un single
precision del registro dado*/
// Funcion que lee dos registros del tipo: Holding Register (HR) cuyo datos son
Float de 32bits....
static void Leer_HR_F32(STR_ALGORITMO_COMUNI *self,FA_DIRECCIONES direccion){

	self->_mb->requestHandler.firstAddr = direccion;
	self->_mb->requestHandler.totalData = 2;
	self->_mb->requestHandler.functionCode = MB_FUNC_READ_HOLDINGREGISTERS;
	self->_mb->requestHandler.slaveAddress = ID_DEL_FILTRO_ACTIVO;
	self->_mb->requestHandler.generate(self->_mb, 0x00, 0x00);

	//forzamos el estado a request
	self->_mb->state=MBM_REQUEST;

 }

////////////////////////////////////////////////////////////////////////////////
//////////////////////777
 /* Funcion PRIMITIVA que generea la peticion en modbus master para leer un single
precision del registro dado*/
// Funcion que lee dos registros del tipo: Holding Register (HR) cuyo datos son
Float de 32bits....
static void Leer_HR_U16(STR_ALGORITMO_COMUNI *self,FA_DIRECCIONES direccion){

	self->_mb->requestHandler.firstAddr = direccion;
	self->_mb->requestHandler.totalData = 1;
	self->_mb->requestHandler.functionCode = MB_FUNC_READ_HOLDINGREGISTERS;
	self->_mb->requestHandler.slaveAddress = ID_DEL_FILTRO_ACTIVO;
	self->_mb->requestHandler.generate(self->_mb, 0x00, 0x00);

	//forzamos el estado a request
	self->_mb->state=MBM_REQUEST;

}
////////////////////////////////////////////////////////////////////////////////
//////////////////////////
 /* Funcion PRIMITIVA que genera la petición en modbus máster para ver si esta
conectado el puertos serie*/
//  para ello hacemos la lectura del ID_Esclavo del Filtro y ver que nos el vsalor
adecuado ID==1
static void Test_Conexion_Canal_MODBUS_OK(STR_ALGORITMO_COMUNI *Obj_COMUNI,int
ID_Esclavo){

	unsigned int Dato_ID_Leido;

		// Esta en el ESTADO de Esperando una Peticion en el MODBUS ...
		if (Obj_COMUNI->_mb->mensaje == MBM_ESPERANDO) {
			// Peticion de Lectura al MODBUS ...
			Obj_COMUNI->PRIMI_Read_HR_U16(Obj_COMUNI, FA_ID_NODO_Slave);
		}
```

```
            //indica que la capa del MODBUS ha recibido el mensaje OK de la
petición...
            if(Obj_COMUNI->_mb->mensaje == MBM_TERMINADO){
                  // Datos recibidos ...
                  Dato_ID_Leido = Obj_COMUNI->_mb->dataResponse.content[1] *256 +
\
                                                Obj_COMUNI->_mb-
>dataResponse.content[2];

                  // Comprobar si lo leido == Identificador Filtro (==1) ....
                  if (Dato_ID_Leido == ID_Esclavo){
                        Obj_COMUNI->Flg_MEF.Flg_COMUNI_Canal_OK = 1;
                  } else Obj_COMUNI->Flg_MEF.Flg_COMUNI_Canal_OK = 0;

                  //forzamos el estado a start para que emita el mensaje
esperando y avanzara la gestion de la comunicacion
                  Obj_COMUNI->_mb->state = MBM_START;

            }

}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////
 /* Funcion PRIMITIVA que generea la peticion en modbus master para Escribir un
single precision del registro dado*/
// Funcion que escribe dos registros del tipo: Holding Register cuyo datos son
Float de 32bits....
// PArametro valor es una buffer de 2 uint16  => Buf[0]=0x0012; Buf[1]=0x0023;
static void Escribir_HR_F32(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES direccion,
unsigned int *valor){


      self->_mb->requestHandler.firstAddr = direccion;
      self->_mb->requestHandler.totalData = 2;
      self->_mb->requestHandler.functionCode = MB_FUNC_WRITE_NREGISTERS;
      self->_mb->requestHandler.slaveAddress = ID_DEL_FILTRO_ACTIVO;
      self->_mb->requestHandler.generate(self->_mb, 0x00, valor );
      self->_mb->state=MBM_REQUEST;

 }

////////////////////////////////////////////////////////////////////////////////
///////////////////////////////
/* Funcion PRIMITIVA que generea la peticion en modbus master para Escribir un
unsigned 16 bits del registro dado*/
// Funcion que escribe un registro del tipo: Holding Register cuyo datos son
EUnisdned int de 16 bits..
static void Escribir_HR_U16(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES direccion,
unsigned int valor){

      self->_mb->requestHandler.firstAddr = direccion;
      self->_mb->requestHandler.totalData = 1;
      self->_mb->requestHandler.functionCode = MB_FUNC_WRITE_HOLDINGREGISTER;
      self->_mb->requestHandler.slaveAddress = ID_DEL_FILTRO_ACTIVO;
      self->_mb->requestHandler.generate(self->_mb, valor, 0x00 );
```

```
	self->_mb->state=MBM_REQUEST;

}

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
// Proceso que controla los estados de las Peticiones y Respuestas del Modbus,
cuando necesitamos hacer
// una lectura de algun parametro del Filtro Activo ....
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////
static int Proceso_Ctrl_Lectura_Modbus_HR_U16(STR_ALGORITMO_COMUNI *Obj_COMUNI,
FA_DIRECCIONES direccion, unsigned int *Almacena_Dato){
	unsigned int Dato_Leido_HR_U16;
	static int dev=0;

	// Esta en el ESTADO de Esperando una Peticion en el MODBUS ...
	if (Obj_COMUNI->_mb->mensaje==MBM_ESPERANDO) {
			// Peticion de Lectura del Registro: "xxxxx" al MODBUS
...
			Obj_COMUNI->PRIMI_Read_HR_U16(Obj_COMUNI, direccion);
			dev=1;
	}

	//indica que la capa del MODBUS ha recibido el mensaje OK de la petición...
	if(Obj_COMUNI->_mb->mensaje==MBM_TERMINADO){
			// Datos recibidos ...
			Dato_Leido_HR_U16 = Obj_COMUNI->_mb-
>dataResponse.content[1] *256 + \
                                                Obj_COMUNI->_mb-
>dataResponse.content[2];

			// Almacenar el dato en la Zona de datos Compartida ...
			//ESTRUCTURA_Compartida_FA.Lecturas.Direct_ON_OFF_u16 =
Dato_Leido;
			//Almacena_Dato->Lecturas.Direct_ON_OFF_u16 =
Dato_Leido_HR_U16;
			*Almacena_Dato = Dato_Leido_HR_U16;

			// Forzamos el estado a start para que emita el mensaje
esperando y avanzara la gestion de la comunicacion
			Obj_COMUNI->_mb->state = MBM_START;
			dev=2;
	}

	return(dev);
}

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
// Proceso que controla los estados de las Peticiones y Respuestas del Modbus,
cuando necesitamos hacer
// una lectura de algun parametro del Filtro Activo ....
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////
```

```
static int Proceso_Ctrl_Lectura_Modbus_HR_F32(STR_ALGORITMO_COMUNI *Obj_COMUNI,
FA_DIRECCIONES direccion, float *Almacena_Dato){
	float Dato_Leido_HR_F32;
	static int dev=0;
	union_dato32 Union_dato;

	// Esta en el ESTADO de Esperando una Peticion en el MODBUS ...
	if (Obj_COMUNI->_mb->mensaje==MBM_ESPERANDO) {
			// Peticion de Lectura del Registro: "xxxxx" al MODBUS
...
			Obj_COMUNI->PRIMI_Read_HR_F32(Obj_COMUNI, direccion);
			dev=1;
	}

	//indica que la capa del MODBUS ha recibido el mensaje OK de la petición...
	if(Obj_COMUNI->_mb->mensaje==MBM_TERMINADO){
			// Datos recibidos ...
			//el primer float llega en la posicion 1 para cada float
(total data =2) siendo los registros de 16bits, en el vector response llegan 4
bytes
			Union_dato.en_2_uint16.dat2.en_2_bytes.byte1 =
Obj_COMUNI->_mb->dataResponse.content[4];
			Union_dato.en_2_uint16.dat2.en_2_bytes.byte2= Obj_COMUNI-
>_mb->dataResponse.content[3];
			Union_dato.en_2_uint16.dat1.en_2_bytes.byte1 =
Obj_COMUNI->_mb->dataResponse.content[2];
			Union_dato.en_2_uint16.dat1.en_2_bytes.byte2= Obj_COMUNI-
>_mb->dataResponse.content[1];

			Dato_Leido_HR_F32 = Union_dato.en_float32;

			// Almacenar el dato en la Zona de datos Compartida ...
			*Almacena_Dato = Dato_Leido_HR_F32;

			// Forzamos el estado a start para que emita el mensaje
esperando y avanzara la gestion de la comunicacion
			Obj_COMUNI->_mb->state = MBM_START;
			dev=2;
	}

	return(dev);
}

////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
// Proceso que controla los estados de las Peticiones y Respuestas del Modbus,
cuando necesitamos hacer
// una lectura de algun parametro del Filtro Activo ....
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////
static int Proceso_Ctrl_Escritura_Modbus_HR_U16(STR_ALGORITMO_COMUNI *Obj_COMUNI,
FA_DIRECCIONES direccion, unsigned int *Escribir_Dato_U16){
	static int dev=0;

	// Esta en el ESTADO de Esperando una Peticion en el MODBUS ...
```

```
	if (Obj_COMUNI->_mb->mensaje==MBM_ESPERANDO) {
				// Peticion de Lectura del Registro: "xxxxx" al MODBUS
...
				Obj_COMUNI->PRIMI_Write_HR_U16(Obj_COMUNI, direccion,
*Escribir_Dato_U16);
				dev=1;
	}

	//indica que la capa del MODBUS ha recibido el mensaje OK de la petición...
	if(Obj_COMUNI->_mb->mensaje==MBM_TERMINADO){
				// Forzamos el estado a start para que emita el mensaje
esperando y avanzara la gestion de la comunicacion
				Obj_COMUNI->_mb->state = MBM_START;
				dev=2;
	}

	return(dev);
}

////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
// Proceso que controla los estados de las Peticiones y Respuestas del Modbus,
cuando necesitamos hacer
// una lectura de algun parametro del Filtro Activo ....
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////
static int Proceso_Ctrl_Escritura_Modbus_HR_F32(STR_ALGORITMO_COMUNI *Obj_COMUNI,
FA_DIRECCIONES direccion, float *Escribe_Dato_F32){
	static int dev=0;
	unsigned int buf[2];
	union_dato32 Union_dato;

	// Esta en el ESTADO de Esperando una Peticion en el MODBUS ...
	if (Obj_COMUNI->_mb->mensaje==MBM_ESPERANDO) {
				Union_dato.en_float32 = *Escribe_Dato_F32;
				buf[0]=Union_dato.en_2_uint16.dat1.en_uint16;
				buf[1]=Union_dato.en_2_uint16.dat2.en_uint16;

				// Peticion de Lectura del Registro: "xxxxx" al MODBUS
...
				Obj_COMUNI->PRIMI_Write_HR_F32(Obj_COMUNI, direccion,
buf);
				dev=1;
	}

	//indica que la capa del MODBUS ha recibido el mensaje OK de la petición...
	if(Obj_COMUNI->_mb->mensaje==MBM_TERMINADO){
				// Forzamos el estado a start para que emita el mensaje
esperando y avanzara la gestion de la comunicacion
				Obj_COMUNI->_mb->state = MBM_START;
				dev=2;
	}

	return(dev);
}
```

```
////////////////////////////////////////////////////////////////////////////////
//////////////////////////
// Envia las peticiones de Lectura de los parametros simples (es decir, NO
contienen Arrays)...
// Utilizar similar a un PROTOTHREAD, en cada "case" ponemos un return para que NO
sea el proceso bloqueante
////////////////////////////////////////////////////////////////////////////////
//////////////////////////
static int Leer_Parametros_Simples(STR_ALGORITMO_COMUNI *Obj_COMUNI){
	//unsigned int Dato_Leido;
	int val_ret;

		switch (Obj_COMUNI->Parametros_Simple_sub_estado){

////////////////////////////////////////////////////////////////////////////////
//////////////////////////////
			case 0: // LECTURA del comando: "Direct_ON_OFF", en el Filtro
Activo ...

				val_ret = Proceso_Ctrl_Lectura_Modbus_HR_U16(Obj_COMUNI,
FA_Direct_ON_OFF, &ESTRUCTURA_Compartida_FA.Lecturas.Direct_ON_OFF_u16);

				if (val_ret == 2) {
					Obj_COMUNI->Parametros_Simple_sub_estado =
Obj_COMUNI->Parametros_Simple_sub_estado + 1;
				}

				return(1);

	////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////
			case 1: // LECTURA del comando: "ESTADO", en el Filtro Activo
...

				val_ret = Proceso_Ctrl_Lectura_Modbus_HR_U16(Obj_COMUNI,
FA_Estado, &ESTRUCTURA_Compartida_FA.Lecturas.Estado_u16);

				if (val_ret == 2) {
					Obj_COMUNI->Parametros_Simple_sub_estado =
Obj_COMUNI->Parametros_Simple_sub_estado + 1;
				}

				return(1);

	////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////
			case 2: // LECTURA del comando: "THDi", en el Filtro Activo ...

				val_ret = Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI,
FA_Estado, &ESTRUCTURA_Compartida_FA.Lecturas.THDi_f32);

				if (val_ret == 2) {
					Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
```

```
                        }

                        return(1);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                  case 3: // LECTURA del comando: "THDv", en el Filtro Activo ...
                        val_ret = Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI,
FA_THDv, &ESTRUCTURA_Compartida_FA.Lecturas.THDv_f32);

                        if (val_ret == 2) {
                                    Obj_COMUNI->Parametros_Simple_sub_estado =
Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                        }
                        return(1);


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                  case 4: // LECTURA del comando: "PF", en el Filtro Activo ...
                        val_ret = Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI,
FA_PF, &ESTRUCTURA_Compartida_FA.Lecturas.PF_f32);

                        if (val_ret == 2) {
                                    Obj_COMUNI->Parametros_Simple_sub_estado =
Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                        }
                        return(1);


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                  case 5: // LECTURA del comando: "valor % del PF", en el Filtro
Activo ...
                        // Ultimo PARAMETRO por Leer ????
                        val_ret = Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI,
COMP_REACTIVA_ACTIVAR_PCiento,
&ESTRUCTURA_Compartida_FA.Lecturas.P_Cien_Reactiva_u32);

                        if (val_ret == 2) {
                                    Obj_COMUNI-
>Parametros_Simple_sub_estado = 0;
                                                }
                        return(val_ret);


                  ////////////////////////////////////////////
                  // AL final poner todo hecho ==> return (2)


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                  default:   // Caso de haber ningun SUBESTADO ?????
                        break;
```

```
            }

      return(val_ret);

}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
static int Leer_FFT_Arrays(STR_ALGORITMO_COMUNI *Obj_COMUNI, FA_DIRECCIONES
direc_Base_Modbus,float *Array_Lectura_F32){
    int val_ret=0;

      switch (Obj_COMUNI->Cnt_Sub_Sub_estado){

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
                              case 0: // Inicializar ...
                                    Obj_COMUNI->Ind_Array_leido=0;
                                    Obj_COMUNI->Cnt_Sub_Sub_estado = Obj_COMUNI-
>Cnt_Sub_Sub_estado + 1;



      ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                              case 1: // Lectura del comando: "Ix_fft_yyy", en
la ...

                                    val_ret =
Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI,(FA_DIRECCIONES) (direc_Base_Modbus
+(Obj_COMUNI->Ind_Array_leido*20)), &Array_Lectura_F32[Obj_COMUNI-
>Ind_Array_leido]);

                                    if (val_ret == 2) {
                                          Obj_COMUNI->Ind_Array_leido =
Obj_COMUNI->Ind_Array_leido + 1;
                                          if(Obj_COMUNI->Ind_Array_leido
> 25) { // Ha leido los 25 Armonicos-Impares
                                                Obj_COMUNI-
>Cnt_Sub_Sub_estado = 0;
                                                return(val_ret);
                                          }
                                    }
                                    return (val_ret);

      ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////


      ////////////////////////////////////////////////////////////////////////////
////////////////////////////////
                              default:   // Caso de NO haber ningun SUBESTADO
?????
                                    break;
```

```
                        }

      return(val_ret);
}
////////////////////////////////////////////////////////////////////////////////
//////////////////////
static int Leer_Parametros_Arrays(STR_ALGORITMO_COMUNI *Obj_COMUNI){

      int val_ret=0;

                  switch (Obj_COMUNI->Parametros_Array_sub_estado){


////////////////////////////////////////////////////////////////////////////////
//////////////////////////////
                        case 0: // Lectura del comando: "IR_rms_Load", en la
Carga ...

                              val_ret =
Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI, FA_IR_Load_rms,
&ESTRUCTURA_Compartida_FA.Lecturas.IR_rms_Carga_f32);

                              if (val_ret == 2) {
                                    Obj_COMUNI-
>Parametros_Array_sub_estado = Obj_COMUNI->Parametros_Array_sub_estado + 1;
                              }

                              return(val_ret);

      ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 1: // Lectura del comando: "IS_rms_Load", en la
Carga ...

                              val_ret =
Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI, FA_IS_Load_rms,
&ESTRUCTURA_Compartida_FA.Lecturas.IS_rms_Carga_f32);

                              if (val_ret == 2) {
                                    Obj_COMUNI-
>Parametros_Array_sub_estado = Obj_COMUNI->Parametros_Array_sub_estado + 1;
                              }
                              return(val_ret);

      ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 2: // Lectura del comando: "IT_rms_Load", en la
Carga ...

                              val_ret =
Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI, FA_IT_Load_rms,
&ESTRUCTURA_Compartida_FA.Lecturas.IT_rms_Carga_f32);
```

```
                                    if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Array_sub_estado = Obj_COMUNI->Parametros_Array_sub_estado + 1;
                                    }

                                    return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                              case 3: /// Lectura del comando: "IN_rms_Load", en la
Carga ...

                                    val_ret =
Proceso_Ctrl_Lectura_Modbus_HR_F32(Obj_COMUNI, FA_IN_Load_rms,
&ESTRUCTURA_Compartida_FA.Lecturas.IN_rms_Carga_f32);

                                    if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Array_sub_estado = Obj_COMUNI->Parametros_Array_sub_estado + 1;
                                    }
                                    return(val_ret);


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////
                              case 4: // LECTURA del comando: "FFT_Linea_R",...
                                    // 1) Seleccionar la FFT ==> Escribir en el
registro de "FFT-Seleccion"
                                    val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_SEL_FFT_FASE,
&ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16);

                                    if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                                // Incrementar la Seleccion de la FFT
....

      ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 =
ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 + 1;
                                    }
                                    return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                              case 5: // LECTURA del ARRAY del comando:
"FFT_Linea_R",...
```

```
                                        val_ret = Leer_FFT_Arrays(Obj_COMUNI,
FA_ARM1, ESTRUCTURA_Compartida_FA.Lecturas.FFT_IR_Linea_f32);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 6: // LECTURA del comando: "FFT_Linea_S",...
                                        // 1) Seleccionar la FFT ==> Escribir en el
registro de "FFT-Seleccion"
                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_SEL_FFT_FASE,
&ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                                // Incrementar la Seleccion de
la FFT ....

      ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 =
ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 7: // LECTURA del ARRAY del comando:
"FFT_Linea_S",...
                                        val_ret = Leer_FFT_Arrays(Obj_COMUNI,
FA_ARM1, ESTRUCTURA_Compartida_FA.Lecturas.FFT_IS_Linea_f32);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 8: // LECTURA del comando: "FFT_Linea_T",...
                                        // 1) Seleccionar la FFT ==> Escribir en el
registro de "FFT-Seleccion"
                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_SEL_FFT_FASE,
&ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16);

                                        if (val_ret == 2) {
```

```
                                                    Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                                    // Incrementar la Seleccion de
la FFT ....

      ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 =
ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 + 1;
                                    }
                                    return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                          case 9: // LECTURA del ARRAY del comando:
"FFT_Linea_T",...
                                    val_ret = Leer_FFT_Arrays(Obj_COMUNI,
FA_ARM1, ESTRUCTURA_Compartida_FA.Lecturas.FFT_IT_Linea_f32);

                                    if (val_ret == 2) {
                                                    Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                    }
                                    return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                          case 10: // LECTURA del comando: "FFT_Linea_N",...
                                     // 1) Seleccionar la FFT ==> Escribir en el
registro de "FFT-Seleccion"
                                    val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_SEL_FFT_FASE,
&ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16);

                                    if (val_ret == 2) {
                                                    Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                                    // Incrementar la Seleccion de
la FFT ....

      ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 =
ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 + 1;
                                    }
                                    return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                          case 11: // LECTURA del ARRAY del comando:
"FFT_Linea_N",...
                                    val_ret = Leer_FFT_Arrays(Obj_COMUNI,
FA_ARM1, ESTRUCTURA_Compartida_FA.Lecturas.FFT_IN_Linea_f32);

                                    if (val_ret == 2) {
```

```
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                        }
                                        return(val_ret);

      //#########################################################################
###########

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////
                        case 12: // LECTURA del comando: "FFT_FILTRO_R",...
                                        // 1) Seleccionar la FFT ==> Escribir en el
registro de "FFT-Seleccion"
                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_SEL_FFT_FASE,
&ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                                // Incrementar la Seleccion de
la FFT ....

      ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 =
ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 13: // LECTURA del ARRAY del comando:
"FFT_Fitro_R",...
                                        val_ret = Leer_FFT_Arrays(Obj_COMUNI,
FA_ARM1, ESTRUCTURA_Compartida_FA.Lecturas.FFT_IR_Filtro_f32);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 14: // LECTURA del comando: "FFT_Filtro_S",...
                                // 1) Seleccionar la FFT ==> Escribir en el
registro de "FFT-Seleccion"
                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_SEL_FFT_FASE,
&ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16);
```

```
                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                                // Incrementar la Seleccion de
la FFT ....

      ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 =
ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                            case 15: // LECTURA del ARRAY del comando:
"FFT_Filtro_S",...
                                        val_ret = Leer_FFT_Arrays(Obj_COMUNI,
FA_ARM1, ESTRUCTURA_Compartida_FA.Lecturas.FFT_IS_Filtro_f32);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                            case 16: // LECTURA del comando: "FFT_Filtro_T",...
                                        // 1) Seleccionar la FFT ==> Escribir en el
registro de "FFT-Seleccion"
                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_SEL_FFT_FASE,
&ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                                // Incrementar la Seleccion de
la FFT ....

      ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 =
ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                            case 17: // LECTURA del ARRAY del comando:
"FFT_Filtro_T",...
                                        val_ret = Leer_FFT_Arrays(Obj_COMUNI,
FA_ARM1, ESTRUCTURA_Compartida_FA.Lecturas.FFT_IT_Filtro_f32);
```

```
                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 18: // LECTURA del comando: "FFT_Filtro_N",...
                                         // 1) Seleccionar la FFT ==> Escribir en el
registro de "FFT-Seleccion"
                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_SEL_FFT_FASE,
&ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                                // Incrementar la Seleccion de
la FFT ....

      ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 =
ESTRUCTURA_Compartida_FA.Escrituras.FFT_Seleccion_u16 + 1;
                                        }
                                        return(val_ret);

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        case 19: // LECTURA del ARRAY del comando:
"FFT_Linea_N",...
                                        val_ret = Leer_FFT_Arrays(Obj_COMUNI,
FA_ARM1, ESTRUCTURA_Compartida_FA.Lecturas.FFT_IN_Filtro_f32);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Parametros_Simple_sub_estado = Obj_COMUNI->Parametros_Simple_sub_estado + 1;
                                        }
                                        return(val_ret);


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                        default:   // Caso de haber ningun SUBESTADO ?????
                                break;
                }

      return(val_ret);
}

////////////////////////////////////////////////////////////////////////////////
//////////////////////////
```

```
// Envia la peticiones de Lectura de las componentes armonicas de cada una de las
fases...
// en el main,, la funcion: callback_modbus realiza el almacenamiento de datos
recibidos del Filtro-Activo
////////////////////////////////////////////////////////////////////////////////
//////////////////////////
static void Algoritmo_leer(STR_ALGORITMO_COMUNI *Obj_COMUNI){
	unsigned int val_ret=0;


	switch (Obj_COMUNI->Sub_Estado_Lectura){

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
			case 0: // LECTURA de todos los comando: Simples, en el
Filtro Activo ...

				val_ret=Leer_Parametros_Simples(Obj_COMUNI);

				if (val_ret == 2) {
					Obj_COMUNI->Sub_Estado_Lectura =
Obj_COMUNI->Sub_Estado_Lectura + 1;
				}

				return;

	////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
			case 1: // LECTURA de los comando: "ARRAYs", en el Filtro
Activo ...

				val_ret = Leer_Parametros_Arrays(Obj_COMUNI);

				if (val_ret == 2) {
					Obj_COMUNI->Sub_Estado_Lectura = 0;
					Obj_COMUNI->Flg_MEF.Leido = 1;
				}

				return;



	////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
			default:   // Caso de haber ningun SUBESTADO ?????
				break;

	}

}
////////////////////////////////////////////////////////////////////////////////
///

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
```

```
/*
 * Funcion que escribe todos los valores de compensacion del Filtro Activo ...
 */
static void Algoritmo_Escribir(STR_ALGORITMO_COMUNI *Obj_COMUNI){
      unsigned int val_ret=0;


            switch (Obj_COMUNI->Sub_Estado_Escritura){

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

      ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                              case 0: // ESCRITURA del comando: "Direct_ON_OFF",
en el Filtro Activo ...

                                    val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, FA_Direct_ON_OFF,
&ESTRUCTURA_Compartida_FA.Escrituras.Direct_ON_OFF_u16);

                                    if (val_ret == 2) {
                                                Obj_COMUNI-
>Sub_Estado_Escritura = Obj_COMUNI->Sub_Estado_Escritura + 1;
                                    }

                                    return;

      ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                              case 1: // ESCRITURA del comando:
"ACTIVAR_DESEQUILIBRIOS", en el Filtro Activo ...

                                    val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, COMP_DESQUIL_ACTIVAR,
&ESTRUCTURA_Compartida_FA.Escrituras.Compensacion_Desequilibrios_ON_OFF_u16);

                                    if (val_ret == 2) {
                                                Obj_COMUNI-
>Sub_Estado_Escritura = Obj_COMUNI->Sub_Estado_Escritura + 1;
                                    }

                                    return;

      ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                              case 2: // ESCRITURA del comando:
"ACTIVAR_ARMONICOS", en el Filtro Activo ...

                                    val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, COMP_ARM_ACTIVAR,
&ESTRUCTURA_Compartida_FA.Escrituras.Compensacion_Armonica_ON_OFF_u16);

                                    if (val_ret == 2) {
```

```
                                                Obj_COMUNI-
>Sub_Estado_Escritura = Obj_COMUNI->Sub_Estado_Escritura + 1;
                                        }

                                        return;

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                                    case 3: // ESCRITURA del comando:
"ACTIVAR_REACTIVA", en el Filtro Activo ...

                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_F32(Obj_COMUNI, COMP_REACTIVA_ACTIVAR_PCiento,
&ESTRUCTURA_Compartida_FA.Escrituras.P_Cien_Reactiva_f32);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Sub_Estado_Escritura = Obj_COMUNI->Sub_Estado_Escritura + 1;
                                        }
                                        return;


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                                    case 4: // ESCRITURA del comando: "PRIORIDAD de
CARGA", en el Filtro Activo ...

                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_U16(Obj_COMUNI, COMP_PRIORIDAD_CARGA,
&ESTRUCTURA_Compartida_FA.Escrituras.Prioridad_Full_load_u16);

                                        if (val_ret == 2) {
                                                Obj_COMUNI-
>Sub_Estado_Escritura = Obj_COMUNI->Sub_Estado_Escritura + 1;
                                        }
                                        return;


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                                    case 5: // ESCRITURA del comando: "Array Escalado
de Armonicos [25]", en el Filtro Activo ...
                                        // Ultimo PARAMETRO por Escribir ????
                                        val_ret =
Proceso_Ctrl_Escritura_Modbus_HR_F32(Obj_COMUNI, (FA_DIRECCIONES) (COMP_H3
+(Obj_COMUNI->Ind_Array_Escrito*10)),
&ESTRUCTURA_Compartida_FA.Escrituras.Factor_Compensacion_Armonicos_f32[Obj_COMUNI-
>Ind_Array_Escrito]);

                                        if (val_ret == 2) {
                                                Obj_COMUNI->Ind_Array_Escrito =
Obj_COMUNI->Ind_Array_Escrito + 1;
                                                if(Obj_COMUNI-
>Ind_Array_Escrito > 25) { // Ha leido los 25 Armonicos-Impares
```

```
                                                                        Obj_COMUNI-
>Sub_Estado_Escritura = 0;
                                                                        Obj_COMUNI-
>Flg_MEF.Escrito = 1;
                                                        }
                                        }

                                        return;


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////
                                default:   // Caso de haber ningun SUBESTADO ?????
                                        break;

            }

}

////////////////////////////////////////////////////////////////////////////////
//////////////////////////
// Funcion que realiza la optimizacion de las componentes Armonicas
// el resultado se almacena en el array de escalados ....
////////////////////////////////////////////////////////////////////////////////
//////////////////////////

////////////////////////////////////////////////////////////////////////////
// MEF de la Algoritmo de ACCIONES COMUNICACION MODBUS.....
// ESTADOS: ESPERANDO, LECTURA, ESCRIBIR (en los Hold_Register del MODBUS)
////////////////////////////////////////////////////////////////////////////
static void Algoritmo_MEF_COMUNI(STR_ALGORITMO_COMUNI *Obj_COMUNI){

      switch(Obj_COMUNI->_Estado){

                                //en este estado se realizan todas las lecturas de
los registros
                                case PROCESO_ESPERANDO:
                                        //self->Flg_MEF.ACCI_LEER=1;
                                        //self->Flg_MEF.ACCI_ESCRIBIR=0;
                                        //self->Flg_MEF.Leido=0;
                                        //self->Flg_MEF.Escrito=0;
                                        Test_Conexion_Canal_MODBUS_OK(Obj_COMUNI,1);

                                        if (Obj_COMUNI-
>Flg_MEF.Flg_COMUNI_Canal_OK==1){
                                                // Acciones que se ponen desde otro
Objeto ( en este caso Objt_OPTI)...
                                                if (Obj_COMUNI->Flg_MEF.ACCI_ESCRIBIR
==1) Obj_COMUNI->_Estado = PROCESO_ESCRIBIR;
                                                else if (Obj_COMUNI-
>Flg_MEF.ACCI_LEER ==1) Obj_COMUNI->_Estado = PROCESO_LEER;

                                                Obj_COMUNI-
>Flg_MEF.Flg_COMUNI_Canal_OK = 0;
```

```
                                        }
                                        break;

                                        //en este estado se realizan todas las
lecturas de los registros
                                case PROCESO_LEER:
                                        Obj_COMUNI->_Operacion_Estado =
LECTURAS_PROCESO ;

                                        Algoritmo_leer(Obj_COMUNI);

                                        if (Obj_COMUNI->Flg_MEF.Leido ==1){
                                                Obj_COMUNI->Flg_MEF.ACCI_LEER=0;
                                                Obj_COMUNI->_Operacion_Estado =
LECTURAS_COMPLETADAS ;
                                                Obj_COMUNI-
>_Estado=PROCESO_ESPERANDO;
                                        }
                                        break;


                                        //en este estado se graban las consignas de
compensacion en el filtro
                                case PROCESO_ESCRIBIR:
                                        Obj_COMUNI->_Operacion_Estado =
ESCRITURA_PROCESO ;

                                        Algoritmo_Escribir(Obj_COMUNI);

                                        if (Obj_COMUNI->Flg_MEF.Escrito ==1) {
                                                Obj_COMUNI->Flg_MEF.ACCI_ESCRIBIR=0;
                                                Obj_COMUNI->_Operacion_Estado =
ESCRITURA_COMPLETADA ;
                                                Obj_COMUNI-
>_Estado=PROCESO_ESPERANDO;
                                        }
                                        break;

                                default:
                                        break;
                        }


}
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
// Inicializacion del OBJETO ALGORITMO de COMUNICACION MODBUS ...
// Parametros:Estrucura Propia y la conexion con el OBJETO del ModbusMaster
void Construct_Algoritmo_COMUNI (STR_ALGORITMO_COMUNI *self, ModbusMaster *mb){

        self->_Estado = PROCESO_ESPERANDO; //empezamos Esperando

        self->_Direcciones =  ARRAY_DIRECCIONES;
```

```
	self->Flg_MEF.ACCI_LEER=1;
	self->Flg_MEF.ACCI_ESCRIBIR=0;
	self->Flg_MEF.Leido=0;
	self->Flg_MEF.Escrito=0;

	self->_Operacion_Estado = LECTURAS_PROCESO;

	self->_mb = mb;

	Construct_Timer_obj(&self->temporizador);

	// void(*MEF_ALGORIT)(STR_ALGORITMO_COMUNI *);
	//static void Algoritmo_MEF_COMUNI(STR_ALGORITMO_COMUNI *self)
	self->MEF_ALGORIT = Algoritmo_MEF_COMUNI;

	// static void Leer_HR_F32(STR_ALGORITMO_COMUNI *self,FA_DIRECCIONES
direccion)
	self->PRIMI_Read_HR_F32 = Leer_HR_F32;
	self->PRIMI_Read_HR_U16 = Leer_HR_U16;
	self->PRIMI_Write_HR_F32 = Escribir_HR_F32;
	self->PRIMI_Write_HR_U16 = Escribir_HR_U16;

	self->Estado_Algoritmo_ini = 1; //indicamos que lo acabamos de CREAR el
OBJETO ...

	self->Parametros_Simple_sub_estado=0;
	self->Parametros_Array_sub_estado=0;

	self->Cnt_Sub_Sub_estado=0;   //Contador de los sub-subestados (bucle
anidados de case...)
	self->Ind_Array_leido=0;   // Indice del array de la Lecturas de las FFT
seleccionadas.....
	self->Ind_Array_Escrito=0;   // Indice del array de la Escritura del Arrays
ESCALADOS_ARMONICOS.....
	self->Sub_Estado_Lectura=0; // variable global del proceso de
Algoritmo_leer().....
	self->Sub_Estado_Escritura=0; // variable global del proceso de
Algoritmo_Escribir().....

}

//////////////////////////////////////////
// FIN del Codigo del Fichero !!!
//////////////////////////////////////////
```

CÓDIGO ALGORITMO COMUNICACION_MODBUS.H

```
#ifndef ALGORITMO_COMUNICACION_MODBUS_H_
#define ALGORITMO_COMUNICACION_MODBUS_H_

////////////////////////////////////////////////////////////////////////////
#include "Timer_obj.h"
#include "ModbusMaster.h"

#define ID_DEL_FILTRO_ACTIVO 1  // Direccion del NODO Modbus ubicado el
Filtro_Activo

#define ULTI_ARMONICO_OPTI   21  // Ultimo armonico a OPTIMIZAR ...
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
#define ACTI_ON   1
#define ACTI_OFF  0

////////////////////////////////////////////////////////////////////////////
#define F_R 0  // Indicador indice del Array en la Lectura/Escritura de Datos ...
#define F_S 1
#define F_T 2
#define F_N 3

#define N_Fases 4
#define N_Armonicos 50

////////////////////////////////////////////////////////////////////////////
////
typedef struct STR_ALGORITMO_COMUNI STR_ALGORITMO_COMUNI;

typedef enum { // ESTADOS de la MEF del Algoritmo COMUNICACIONES....
      PROCESO_ESPERANDO,
      PROCESO_LEER,
      PROCESO_ESCRIBIR
}STR_ALGORITMO_ESTADO;

typedef enum{ // Numero de Fase de Lectura...
      FA_FASE_IR = 0,
      FA_FASE_IS = 1,
      FA_FASE_IT = 2,
      FA_FASE_IN = 3
}FA_FASES;

// Direcciones_MODB US del Filtro_Activo a LEER/ESCRITURA ...
// Ver manual del Filtro Schnaffer...
typedef enum {
      FA_Ninguna =-1,
      FA_Estado = 200,
      FA_Direct_ON_OFF =2020,
```

```
FA_THDi = 1010,
   FA_PF = 1020,
   FA_THDv = 1190,

   FA_IR_Load_rms = 1300,
   FA_IS_Load_rms = 1310,
   FA_IT_Load_rms = 1320,
   FA_IN_Load_rms = 1480,

   FA_ON_OFF = 2020,

   FA_ID_NODO_Slave = 2300,

   COMP_REACTIVA_ACTIVAR_PCiento = 4000,
   COMP_REACTIVA_PF_Referencia = 4010,

   COMP_DESQUIL_ACTIVAR = 4050,

   COMP_PRIORIDAD_CARGA = 4070,

   COMP_ARM_ACTIVAR = 4100,

COMP_H3 = 4110,
COMP_H5 = 4120,
COMP_H7 = 4130,
COMP_H9 = 4140,
COMP_H11 = 4150,
COMP_H13 = 4160,
COMP_H15 = 4170,
COMP_H17 = 4180,
COMP_H19 = 4190,
COMP_H21 = 4200,
COMP_H23 = 4210,
COMP_H25 = 4220,

COMP_H27 = 4230,
COMP_H29 = 4240,
COMP_H31 = 4250,
COMP_H33 = 4260,
COMP_H35 = 4270,
COMP_H37 = 4280,
COMP_H39 = 4290,
COMP_H41 = 4300,


COMP_H43 = 4310,
COMP_H45 = 4320,
COMP_H47 = 4330,
COMP_H49 = 4340,

   FA_SEL_FFT_FASE = 7000,
          FA_ARM1 = 7010,
          FA_ARM2 = 7020,
          FA_ARM3 = 7030,
          FA_ARM4 = 7040,
```

```
		FA_ARM5 = 7050,
		FA_ARM6 = 7060,
		FA_ARM7 = 7070,
		FA_ARM8 = 7080,
		FA_ARM9 = 7090,
		FA_ARM10 = 7100,
		FA_ARM11 = 7110,
		FA_ARM12 = 7120,
		FA_ARM13 = 7130,
		FA_ARM14 = 7140,
		FA_ARM15 = 7150,
		FA_ARM16 = 7160,
		FA_ARM17 = 7170,
		FA_ARM18 = 7180,
		FA_ARM19 = 7190,
		FA_ARM20 = 7200,
		FA_ARM21 = 7210,
}FA_DIRECCIONES;

// Dentro del ESTADO-Lecturas indica si esta en proceso de Lectura o se han
finalizado...
typedef enum {
	LECTURAS_PROCESO,
	LECTURAS_COMPLETADAS,
	ESCRITURA_PROCESO,
	ESCRITURA_COMPLETADA
}OPERACION_ESTADO;

typedef struct {
			unsigned ACCI_LEER:1;
			unsigned ACCI_ESCRIBIR:1;
			unsigned Leido:1;
			unsigned Escrito:1;
			unsigned Flg_COMUNI_Canal_OK:1;
			unsigned bit5:1;
			unsigned bit6:1;
			unsigned bit7:1;
			unsigned bit8:1;
			unsigned bit9:1;
			unsigned bit10:1;
			unsigned bit11:1;
			unsigned bit12:1;
			unsigned bit13:1;
			unsigned bit14:1;
			unsigned bit15:1;
} FLAGS_COMUNICACION;

////////////////////////////////////////////////////////////////////////////
// Estructura del ALGORITMO DE OPTIMIZACION DE POTENCIA DEL CONVERTIDOR
////////////////////////////////////////////////////////////////////////////
struct STR_ALGORITMO_COMUNI {

	STR_ALGORITMO_ESTADO _Estado;
                                                    // Hay que tner en cuenta que
se inicia en el valor cero y luego se incrementa
```

```
        FLAGS_COMUNICACION Flg_MEF;

        unsigned int * _Direcciones; // Ptero a las direcciones de acceso (Lectura)
a los datos del Modbus
        OPERACION_ESTADO _Operacion_Estado;
        Timer temporizador;   //para controlar el tiempo de espera para la respuesta
de las fft's en el filtro al cambiar de fases

        void(*MEF_ALGORIT)(STR_ALGORITMO_COMUNI *);  // Puntero a la funcion de
ejecucion de la MEF del Propio Algoritmo...

        // PRIMITIVAS DE LECTURA /ESCRITURA en el bus MODBUS ....
        void(*PRIMI_Read_HR_F32)(STR_ALGORITMO_COMUNI *,FA_DIRECCIONES direccion);
// Puntero a la funcion de ejecucion de la MEF del Propio Algoritmo...
        void(*PRIMI_Read_HR_U16)(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES
direccion);
        void(*PRIMI_Write_HR_F32)(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES
direccion, unsigned int *valor);
        void(*PRIMI_Write_HR_U16)(STR_ALGORITMO_COMUNI *self, FA_DIRECCIONES
direccion, unsigned int valor);
        ////////////////////////////////////////////////////////////////////////
///////
        ModbusMaster *_mb; //puntero al modbus master

        unsigned int Estado_Algoritmo_ini;  // Estado que indica que ha pasado al
menos una vez por el Algoritmo...

        unsigned int Parametros_Simple_sub_estado;  //variable para controlar el
estado de las lecturas del los parametros simples (NO ARRAYs)
        unsigned int Parametros_Array_sub_estado;   //variable para controlar el
estado de las lecturas de los ARRAYs ...
        unsigned int Cnt_Sub_Sub_estado;   //Contador de los sub-subestados (bucle
anidados de case...)
        unsigned int Ind_Array_leido;   // Indice del array de la Lecturas de las
FFT seleccionadas.....
        unsigned int Ind_Array_Escrito;   // Indice del array de la Escritura del
Arrays ESCALADOS_ARMONICOS.....

        unsigned int Sub_Estado_Lectura;   // variable global del proceso de
Algoritmo_leer().....
        unsigned int Sub_Estado_Escritura;   // variable global del proceso de
Algoritmo_Escribir().....
};

////////////////////////////////////////////////////////////////////////////////
// Constructor: Inicializar la Estructura del Algortimo COMUNICACIONES....
void Construct_Algoritmo_COMUNI(STR_ALGORITMO_COMUNI *self, ModbusMaster *mb);

// Definición de funciones PROTIPOS en este FicherO:
ALGORITMO_COMUNICACION_MODBUS.c
static void Algoritmo_leer(STR_ALGORITMO_COMUNI *self);
static void Algoritmo_Escribir(STR_ALGORITMO_COMUNI *self);


    #endif /* INCLUDE_ALGORITMO_COMUNICACION_MODBUS_H */
```

CÓDIGO ALGORITMO STC_ COMPARTIDA_FILTRO_ACTIVO.C

Estructura de datos compartida entre la MEF Optimizador y MEF Comunicaciones.

```
#include "STR_Compartida_Filtro_Activo.h"

STR_Compartida_FA ESTRUCTURA_Compartida_FA;
```

CÓDIGO ALGORITMO STC_ COMPARTIDA_FILTRO_ACTIVO.C

```
#ifndef __STR_Compartida_Filtro_Activo_H__
#define __STR_Compartida_Filtro_Activo_H__


typedef struct  {

            unsigned int ID_u16;
            unsigned int Estado_u16;
            unsigned int Direct_ON_OFF_u16;

            float THDi_f32;
            float THDv_f32;
            float PF_f32;
            float P_Cien_Reactiva_u32;

            float IR_rms_Carga_f32;
            float IS_rms_Carga_f32;
            float IT_rms_Carga_f32;
            float IN_rms_Carga_f32;

            float IR_rms_Filtro_f32;
            float IS_rms_Filtro_f32;
            float IT_rms_Filtro_f32;
            float IN_rms_Filtro_f32;

            float FFT_IR_Linea_f32[25];
            float FFT_IS_Linea_f32[25];
            float FFT_IT_Linea_f32[25];
            float FFT_IN_Linea_f32[25];

            float FFT_IR_Filtro_f32[25];
            float FFT_IS_Filtro_f32[25];
            float FFT_IT_Filtro_f32[25];
            float FFT_IN_Filtro_f32[25];
```

```
}Leer;


typedef struct {
                    unsigned int Direct_ON_OFF_u16;
                    unsigned int Compensacion_Armonica_ON_OFF_u16;
                    unsigned int Compensacion_Desequilibrios_ON_OFF_u16;


                    unsigned int Prioridad_Full_load_u16;
                    unsigned int FFT_Seleccion_u16;

                    float Cos_fi_Reactiva_f32;
                    float P_Cien_Reactiva_f32;
                    float Factor_Compensacion_Armonicos_f32[25];

}Escribir;

typedef struct {
        Leer Lecturas;
        Escribir Escrituras;
}STR_Compartida_FA;

extern STR_Compartida_FA ESTRUCTURA_Compartida_FA;

#endif /* __STR_Compartida_Filtro_Activo_H__ */

////////////////////////////////////////////////////////////////////////////
// FIN DEL CODIGO DEL FICHERO
////////////////////////////////////////////////////////////////////////////
```

ÍNDICE DE TABLAS