電子ビーム・プラズマ系における非線形現象

竹田　剛

静岡大学大学院　理工学研究科　物質科学専攻

Preface

真空放電に見られる電離した気体に対し、固体、液体、気体とは著しく異なった第４の物質状態としての認識から、1928 年に Langmuir はプラズマ (plasma) と命名している。以来、プラズマ物理学は急速に関心を集め、多方面の分野に影響を与え続けている。現在、文明社会におけるプラズマは、宇宙空間の 99 %以上の物質における存在形態として理解されているだけでなく、積極的に工学利用されている。

1950 年代前半から、非線形項をもつ方程式に対し、コンピュータを駆使してシミュレートする手法が Fermi らにより採用されはじめ、1955 年の Fermi-Pasta-Ulam の再帰現象の発見が、非線形分野での大きな功績となっている。このようにして、コンピュータによる数値解析は、これまで敬遠されていた複雑な非線形方程式に対し、数値解だけでなく新しい知見をも与えている。以後、プラズマ中における非線形波動の理論的、実験的研究が活発となり、非線形波動方程式などから予見されていたソリトン (soliton) や変調不安定性 (modulational instability) などの存在が、実験的に観測されている。近年、物理学者と数学者を中心とした非線形の研究が、他の学問分野の研究においても重要視されている。

プラズマ中に電子ビームを入射する電子ビーム・プラズマ系においては、変調された波の塊を意味する「波束 (wave packet)」が励起されることが知られている。電子の応答時間スケールで発展する波束は、振幅の発展に成長、飽和と減衰の三つの過程を持ち、成長過程では線形方程式で記述されるビーム・モード特性に従うことが分かっている。しかし、飽和過程と減衰過程については、いまだ、多くの研究と議論が必要とされ、非線形現象の観測に注目が集まっている。本研究では、この系における非線形現象を探るべく、電子の応答時間スケールで空間発展する波束に対し、同軸プローブとエネルギー・アナライザーを用いて、ポテンシャルと電子ビームの速度分布関数の実験的観測を行った。同軸プローブによる観測から、波束についての成長、飽和、減衰からなる発展過程が示された。エネルギー・アナライザーによる観測から、プラズマ密度に対しビーム密度が 1.2% の場合、低周波帯域においては、高速側への広がり（タイプ１）と低速側へ伸びた異常な広がり（タイプ２）が示され、高周波帯域においては、時間発展をする渦に似たドーナッツ状の分布が示された。このドーナッツ状の分布の中心の速度と波束の位相速度は一致し、また、ドーナッツ状の分布の速度半径の二乗がポテンシャルの大きさに比例していた。このときのビーム電子を捕捉するポテンシャルの大きさは、最大で約 0.3V と見積もられた。これらの事実は、ドーナッツ

状の分布が、波束のポテンシャルにビーム電子が捕捉されて形成される渦であることを意味している。そして、タイプ２のような分布の形成メカニズムとして、この系で発生する自然励起波のポテンシャルが、ビーム電子に対し捕捉や脱捕捉などの散乱過程をもたらしていることが予想できる。

本論文は、1929 年に Tonks と Langmuir によりプラズマ振動が発見されて以来、発展してきたプラズマ波動の分野において、電子ビーム・プラズマ系における非線形現象についての上記のような実験的研究成果を報告するものである。第 1 章は本研究における基礎理論、研究の背景および目的について、第 2 章は実験システムを構成する装置類、検出プローブおよび測定システムについて、第 3 章と第 4 章は実験結果と議論、そして結論について記述している。なお、観測に用いた装置類の電子回路図と制御プログラム、さらに、観測データを解析するためのプログラムを末巻に記載している。

本研究および本論文の起草にあたり、教示および忠告を下さった指導教官の山際啓一郎教授に感謝の意を捧げたい。また、有益な助言や意見を下さった佐伯紘一教授、飽本一裕助教授 (帝京大学)、装置類の工作で協力して頂いた池谷隆司技官、多方面で支援してくれた山際研究室のゼミ生らに心からお礼を言いたい。

2003 年 6 月

竹田　剛

Contents

Chapter 1

序論

1.1 プラズマの特徴

固体、液体、気体などの物質状態と異なり「第 4 の状態」とも呼ばれるプラズマ (plasma) とは、「荷電粒子を含んだ（ほぼ中性の）粒子集団」を意味している。プラズマ状態においては、無衝突状態にあっても粒子間には電磁力による相互作用が生じ、局所平均電荷を 0 にしようとする強い特性が働いている。プラズマは、真空放電のみならず、電離層、オーロラ (aurora)、バンアレン帯 (van Allen belt)、太陽などにみられる。実験室や宇宙空間で対象となるプラズマは、クーロン (coulomb)・ポテンシャル・エネルギーが、プラズマ粒子の熱運動エネルギー $k_{\mathrm{B}}T$ より小さく ($n{\lambda_{\mathrm{D}}}^3 \gg 1$)、かつ、フェルミ

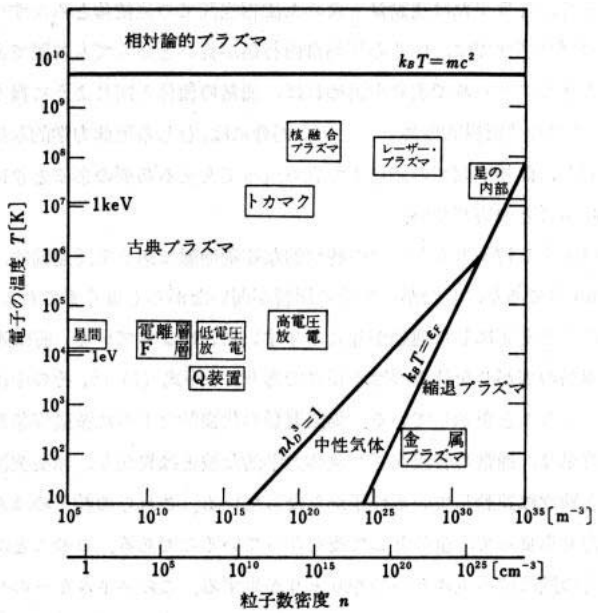


Fig. 1.1: Diagram of plasma states.

(Fermi)・エネルギー $\varepsilon_{\rm F}$ より小さい状態にあるとして、古典的に扱うことができる。ここで、$\lambda_{\rm D} = (\epsilon KT/ne^2)^{1/2}$ はデバイ長 (Debye length) を意味する。Fig.1.1 は、温度 T と粒子密度 n で位置付をした種々のプラズマにおける状態図 [1] を表している。プラズマ粒子の集団運動により生じる波動は、無衝突で独立に個別運動している粒子の軌道の小さなずれの集まりがもたらす現象である。さらに、波と粒子の間に相互作用が生じる場合、ランダウ減衰 (Landau damping)、不安定性 (instability)、非線形波動 (nonlinear wave)、粒子捕捉 (particle trapping) などの現象が顕著になりやすい。

1.2 冷たい電子ビーム・プラズマ波

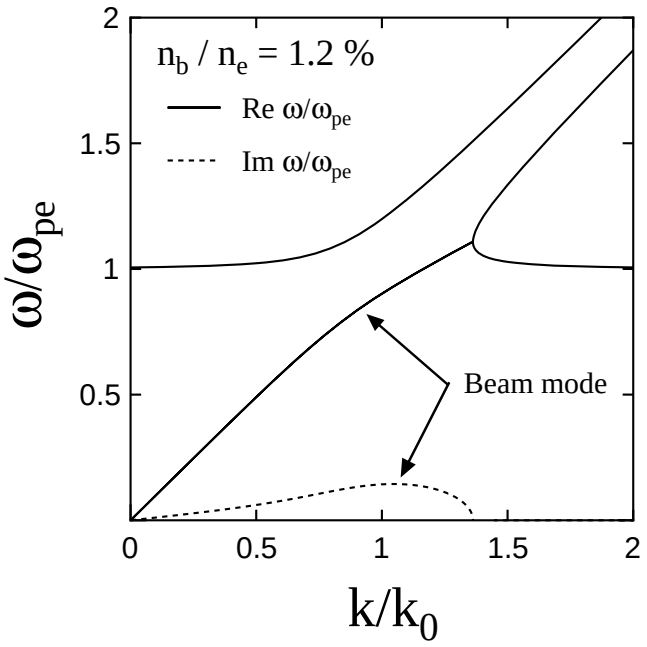


Fig. 1.2: Dispersion relation for cold electron-beam-plasma in the case of $n_{\rm b}/n_{\rm e} = 1.2\%$.

ビーム不安定性における単純な系として、磁場を持たない一様な密度 $n_{\rm e}$ の電子プラズマ中に、単一エネルギーを持つ速度 $v_{\rm b}$、密度 $n_{\rm b}$ の電子ビームが入射した状況を考える。各粒子の温度がゼロのデルタ関数の粒子分布を、縦誘電率方程式に適用し、

$$\frac{n_{\rm b}/n_{\rm e}}{(\omega/\omega_{\rm pe} - k/k_0)^2} + \frac{1}{(\omega/\omega_{\rm pe})^2} = 1 \tag{1.1}$$

で表される冷たい電子ビーム・プラズマ波の分散関係式が導かれる [2,4]。ここで、$\omega_{\rm pe}$ は、電子プラズマ周波数 (electron-plasma frequency) であり、$k_0 \equiv \omega_{\rm pe}/v_{\rm b}$ を便宜的に定義している。Fig. 1.2 に $n_{\rm b} = 1.2\%$ の場合における分散曲線を示す。Re $\omega/\omega_{\rm pe} = 1$ 付近で、時間成長率は最大となることから、式 1.1 を級数展開することで、

$$\max\ {\rm Im}(\omega/\omega_{\rm pe}) = (3)^{1/2}(1/2)^{4/3}(n_{\rm b}/n_{\rm e})^{1/3} \tag{1.2}$$

で表す時間成長率のビーム密度依存性が示される [4]。

1.3 粒子捕捉現象

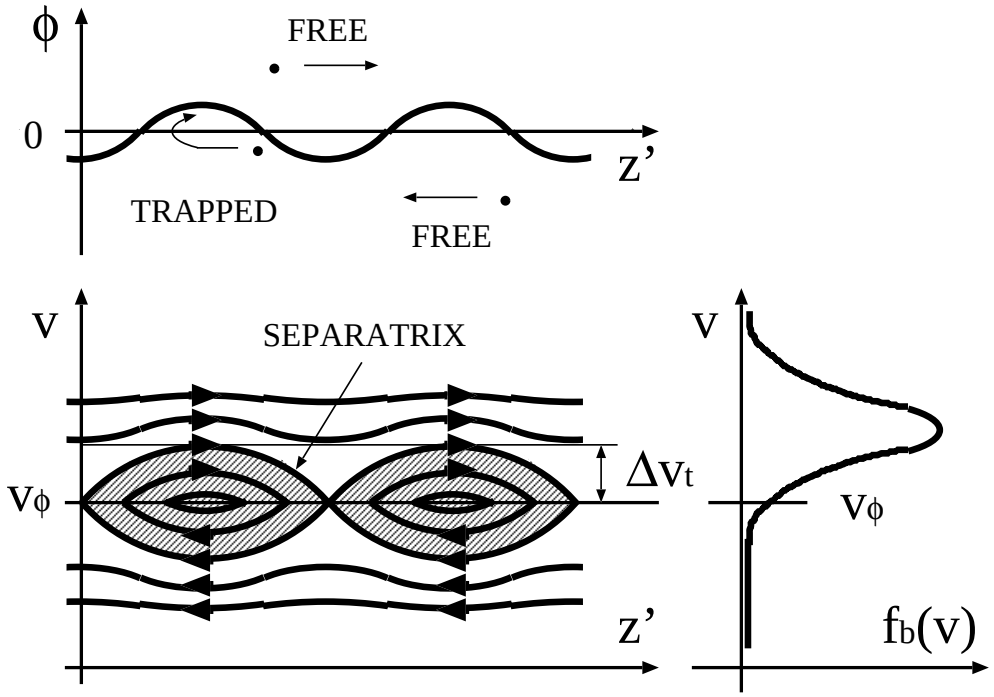


Fig. 1.3: Phase-space trajectories for beam-electrons moving in a potential wave.

不安定波動が時間とともに指数関数的に成長を続ける一方で、波のポテンシャルを越えられない粒子は、ある位相空間内に次々と捕捉され往復運動を始める。粒子が電子であるとすると、Fig.1.3 で示されるような自由電子と捕捉電子の境界となるセパラトリックス（separatrix）は、

$$v - v_\phi = \pm\sqrt{2(1+\sin(kz'))}\omega_\mathrm{t}/k \tag{1.3}$$

で記述できる [19]。ここで、v_ϕ は波の位相速度、ω_t はバウンス周波数 (bounce frequency)、$z' = z - v_\phi t$ は波とともに動く座標系を表す。1.3 における最大値は、Fig.1.3 の捕捉半径 Δv_t と一致することから、$\omega_\mathrm{t}^2 = ek^2\phi/m$ として、

$$\Delta v_\mathrm{t}^2 = \frac{4e\phi}{m} \tag{1.4}$$

が成り立つ [6]。

1.4 本研究の目的

電子ビーム・プラズマ系において波の振幅が大きい場合、代表的な非線形現象 [6, 7] が現れやすいため、例えば、ソリトン [8]、変調不安定性 [9]、粒子捕捉 [2, 6] などについての議論も必要となってくる。Wong [10] らは、パルス状の電子ビームにより励起される 3 次元のラングミュア・ソリトンの崩れを、イオンの応答時間スケールで観測し、Zakharov [11] の理論と一致することを報告している。Intrator [12] らは、電子の応答時間スケールで発展する電子波の波束において、変調不安定性による縦方向の自己収束を観測し、この現象を非線形幾何光学理論 (nonlinear geometrical optics theory) [13] で説明できるとしている。山際 [14] らは、Fig.1.4 で示すように、波束が複数の波束を次々と放出し波束列へと発展することで、波の安定化が得られることを観測している。著者 [15] らは、エネルギー・アナライザーを用いたビームのエネルギー分布関数の観測から、非対称な波束の発生メカニズムに、粒子捕捉現象が関っている可能性を報告している。

本研究は、電子ビーム・プラズマ系における非線形現象として、電子の応答時間スケールで発展する波束に着目し、エネルギー・アナライザーを用いてビームの分布関数を観測することで、山際らが観測した波束の発展メカニズムを解明しようとするものである。

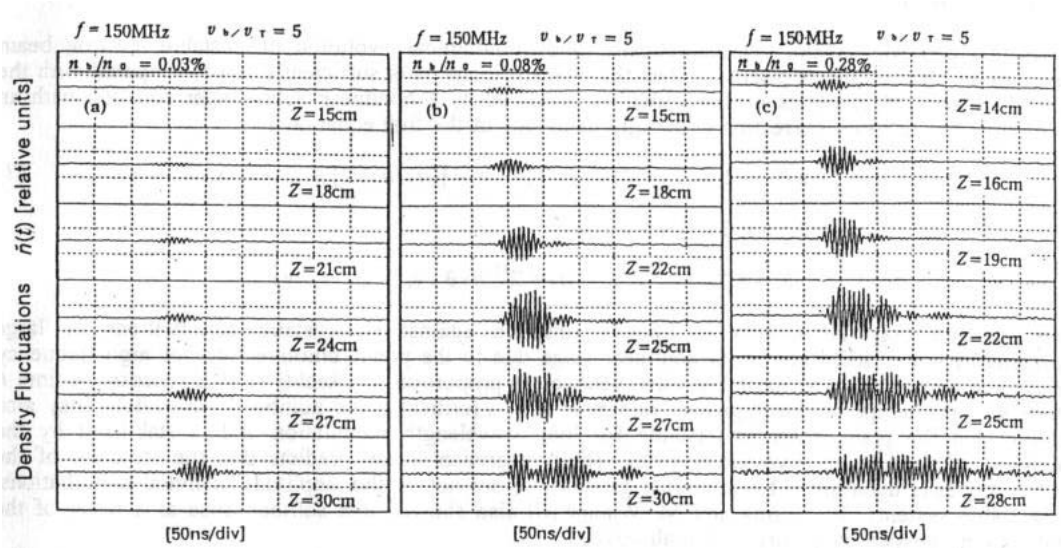


Fig. 1.4: Nonlinear evolution of a single burst wave and emission of a series of burst waves (After Yamagiwa *et al*, 1997).

Chapter 2

実験システム

2.1 電子ビーム・プラズマ装置

プラズマは、アルゴン・ガスで満たされた直径 0.26m、長さ 1.2m のステンレス製の円筒容器内にて、4 本の熱フィラメントと容器壁の間での直流放電により生成される。この 4 本のフィラメントは、タングステン製で、放電時には 2000 度近くに熱せられる。プラズマの生成効率を上げるために、円筒外壁に 12 本、両端外壁に 10 本の Sm-Co 製の永久磁石を設置し、フル・ライン・カスプ (full-line cusp) 磁場 [16] によるプラズマの閉じ込めを行っている。排気システムは、回転ポンプ、拡散ポンプ、チタン・ゲッター・ポンプから成り、容器壁、熱源およびその周辺から発生する不純物中性ガスの排気にも十分に対応できている。ガス圧は、B-A ゲージ（超高真空計）で確認しながら、ニードル・バルブで調節を行っている。なお、ディスク・プローブ (disk probe) を用いて測定されるラングミュア・プローブ特性曲線から、アルゴン・ガス圧 2.8×10^{-5}Torr で、1eV 未満の電子温度を持つ極めて冷たいプラズマが、実現されることを確認している。

電子ビーム・ガン (electron-beam gun)（以下、ビーム・ガン）は、Fig.2.1 の概略図で示すように、主に、熱カソード (hot cathode)、制御グリッド (control grid)、アノード・グリッド (anode grid) で構成され、直径 50mm の開口を持つ浮遊カバー (floating cover) に覆われている。カソードには、ベース・メタルにニッケルを使用し、表面をバリウム・ストロンチウム・カルシウムの 3 元酸化物が覆っている。制御グリッドには、熱カソードからの輻射熱による影響を考慮し、太さ 0.03mm、密度 50mesh/inch のタングステン製のメッシュを使用している。アノード・グリッドには、太さ 0.1mm、密度 25mesh/inch のモリブデン製のメッシュを使用している。タングステン製のフィラメント・ヒーター (filament heater) にて熱せられるカソードの表面温度は、最高 900 度ぐらいまで上げられる。各電極間の絶縁には、高温状態でも内部抵抗および表面抵抗の大きいセラミック (ceramic) とアルミナ (alumina) を用いている。電子ビーム（以下、ビーム）は、制御グリッドの熱カソードに対する電位 $V_{\rm b}$ が正の場合に引き出され、0

の場合にカット・オフとなる。ビームの初期の平均速度は、熱カソードとアノード・グリッドの電位差によって決まるが、冷たいビームの条件 $\Delta v/v_{\rm b} < (n_{\rm b}/n_{\rm e})^{1/3}$ [4] を満たすように、ビームの熱速度に対し十分に速く設定している。また、プラズマ半径の有限性により、ニ流体不安定性が生じるためのビーム速度の上限 [4] は、$\omega_{\rm pe}a/2.4 > v_{\rm b}$ となる。ここで、a はプラズマ半径である。z 軸上の一端に設置されたビーム・ガンから、プラズマ中にビームが入射される際、ビームが一次元的に振る舞い、かつ、そのほとんどがコレクター (collector) へ吸収されるように、6 基のコイルが発生する z 軸方向磁場を円筒容器内に印加している。このとき、電子サイクロトロン周波数が電子プラズマ周波数より大きくなるように強い磁場を設定し、軸方向の波が支配的であると見なすことができる [2]。ビームを収集するコレクターは、直径 50mm のディスク形状を有し同軸ケーブルで接続されている。ビーム・プラズマ装置の実験時における構成を Fig.2.2 に示す。

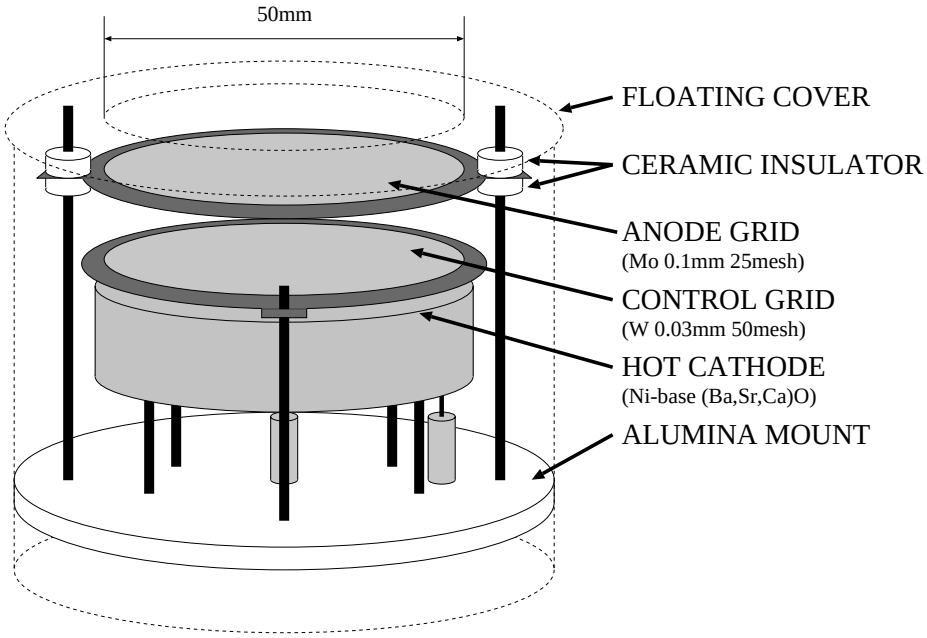


Fig. 2.1: Schematic of electron-beam gun.

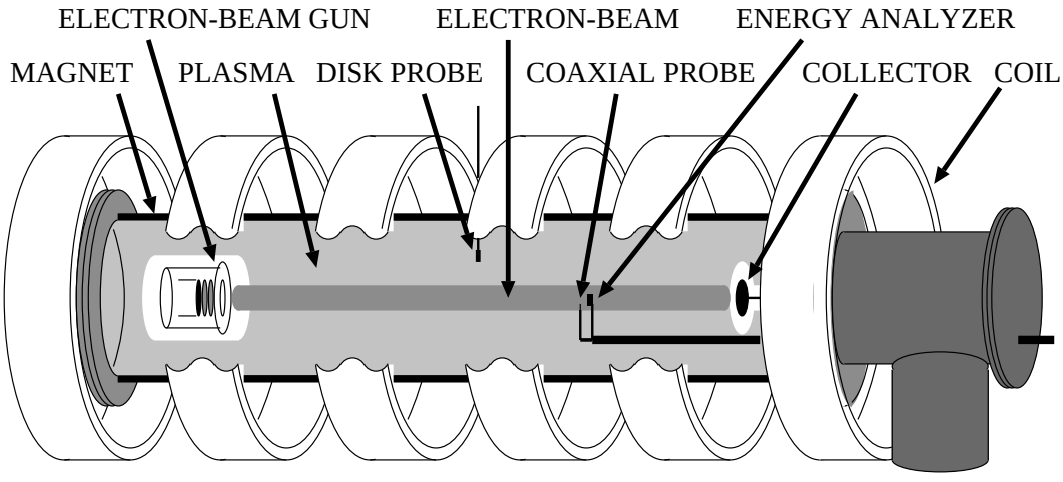


Fig. 2.2: Experimental setup.

2.2 検出プローブ

ビームの速度分布を検出するためのプローブとして、エネルギー・アナライザー (energy analyzer) [17]（以下、アナライザー）を採用している。アナライザーの構造は、Fig.2.3 で示すように、分別グリッド (discriminator grid) と電場に対し遮蔽されたコレクター (collector) から成り、直径 5.5mm の開口を持つセラミック製の外被で囲まれている。分別グリッドと遮蔽グリッドには、太さ 0.03mm、密度 80mesh/inch のモリブデン製のメッシュを使用している。ここでも、各電極間の絶縁にセラミックを使用している。任意の電圧 $V_{\rm D}$(=-75–0V) を与えられた分別グリッドにより、ビームがアナライザーのコレクターへの流入制限を受け、$V_{\rm D}$ に依存するコレクター電流 $I_{\rm c}$ が測定できる。ここで、ビームの分布関数として、一次元のドリフト・マクスウェル分布 (Drift Maxwellian distribution) $f_{\rm b}(v)$ を仮定すると、$I_{\rm c}$ は、

$$-I_{\rm c}(v) = en_{\rm b}\int_{-\infty}^{v} v f_{\rm b}(v){\rm d}v \tag{2.1}$$

で記述され、両辺を一階微分して

$$-\frac{dI_{\rm c}(v)}{dV_{\rm D}} = \frac{e^2 n_{\rm b}}{m} f_{\rm b}(v) \tag{2.2}$$

を得る。ただし、$eV_{\rm D} = mv^2/2$ とする。従って、Fig.2.6 で示すように、特性曲線 $V_{\rm D} - I_{\rm c}$ に対し一階微分を与えることで、ビームの速度分布関数が得られる。一方、ポテンシャルは、同軸プローブ (coaxial probe) にて検出を行っている。このプローブの先端 (tip) は、極細の同軸ケーブルの内軸を 2mm 剥き出しにしたものである。Fig.2.4 の概略図で示されるこれらのプローブは、ガラス・カバーで覆われた直径 1.2mm の同軸ケーブルに接続され、検出信号に対し高周波マッチングがとれるように、ハーメチック・シール (hermetic seal) を挟む大気側で、Fig.2.5 で示すマッチング抵抗が接続されている。また、ガラス・カバーの固定にはトル・シール (torr seal) を使用している。これらのプローブは、ビーム・ガンから 15–60cm の範囲で一体となって移動させることができる。実験の際には、FFT によるスペクトル解析を考慮して、1/3cm 刻みの 128 点の各位置で信号の検出を行っている。

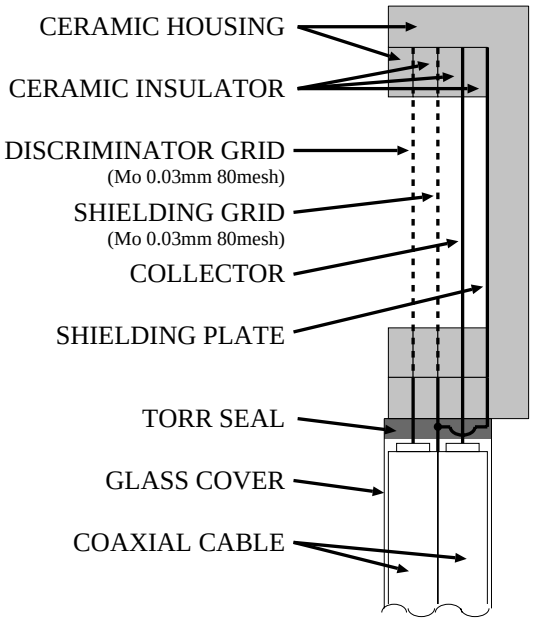


Fig. 2.3: Construction of energy analyzer.

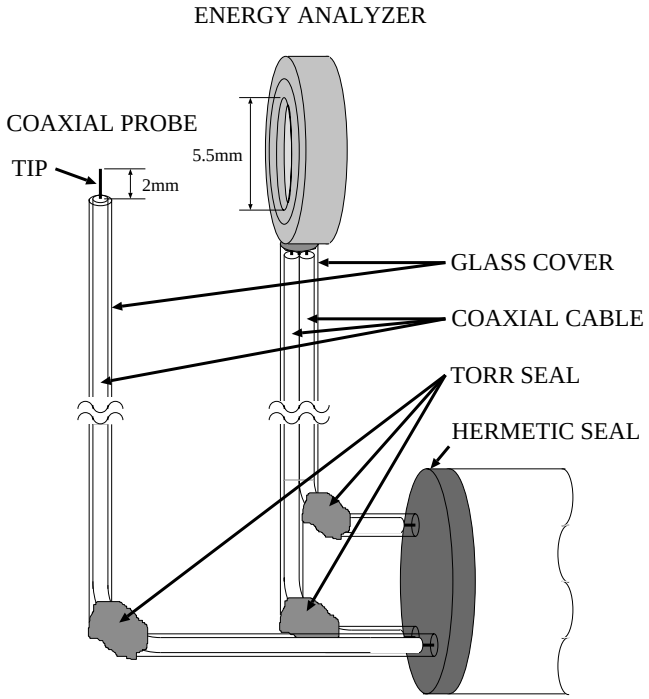


Fig. 2.4: Schematic of energy analyzer and coaxial probe.

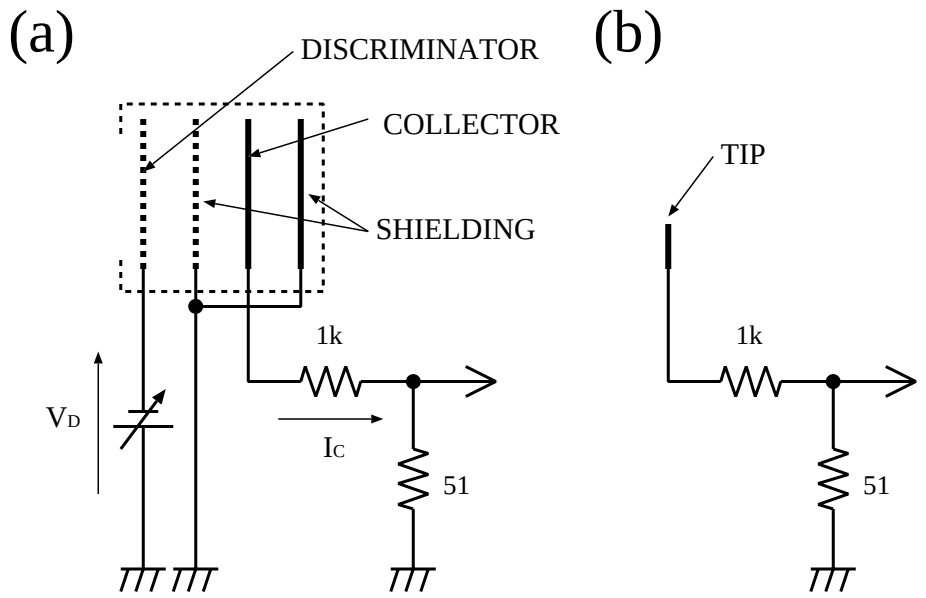


Fig. 2.5: Matching circuits connected; (a) at the end of energy analyzer, and (b) at the end of coaxial probe.

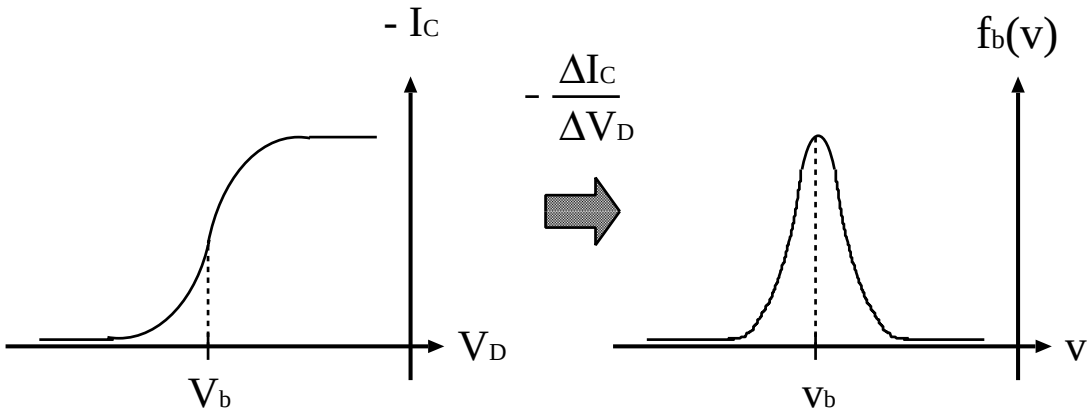


Fig. 2.6: Derivation of distribusion function $f_{\mathrm{b}}(v)$.

2.3 観測システム

実験では、矩形状のパルス (pulse) とテスト波 (test wave) を初期の波束の励起源として用い、波束の発展過程におけるポテンシャルとビームの速度分布関数の観測から非線形現象を捉えている。パルスは、パルス発信器で発生し、3.6μs の時間幅を持っている。テスト波として、RF(70–100MHz) が単一のの正弦波 (18MHz) からの振幅変調 (AM) を受けて形成される単一波束を採用している。このテスト波は、50ns の時間半値幅をもち、パルス時間の中間時刻 (1.8μs) で同期発振している。実験では、Fig.2.7 に示す RF の異なる 4 つのテスト波を使用している。ビームを駆動させるための回路構成を、Fig.2.8 に示す。ビーム・ガンの制御グリッドに、パルスをパルス・トランスを介して、さらに、テスト波をコンデンサーを介して印加している。従って、このときの足し合わせ信号により、波束を搬送するパルス・ビームを駆動させることができる。このとき、パルスの振幅がビーム密度 $n_{\rm b}$ を、加速電圧 $V_{\rm b}$(=-50V) が初期ビームの平均速度 $v_{\rm b}$ を決定している。熱カソードの表面温度は、ヒーター電流 $I_{\rm h}$ で決定され、熱電子が励起できる状態に設定されている。図中において、下方の点線で囲まれた部分はビーム駆動回路 (electron-beam driver) に相当し、テスト波およびパルスに対して高周波マッチングが取れるよう、いくつかの抵抗を接続している。

Fig.2.10 に観測システムの概略を示す。パルスとそれに同期するテスト波は、ビーム・ドライバーを介してビーム・ガンに印加され、初期波束を伴うパルス・ビームがプラズマ中へ入射される。このとき、Fig.2.9 で表すように、アナライザーで検出される信号は 2 つに分配されて、低周波アンプ (DC–8MHz, 46dB) と高周波アンプ (8–1300MHz, 40dB) でそれぞれ増幅される。これらの低周波信号と高周波信号は、メインのデジタル・オシロスコープ（以下、オシロスコープ）の各チャンネルにて受信される。このオシロスコープは、アヴェレージ処理を行うことができ、さらに、1ns の時間分解能 (サンプリング) と 10bit の電圧分解能（階調度）を有しているので、高周波アンプからの信号に含まれるプラズマ電子時間スケールで発展する波束を十分捉えることができる。受信される 2 つの信号データは、GP-IB を介してメインの PC に取り込まれ、実験データとしてハードディスクに記録される。同軸プローブで検出した信号は、高周波アンプのみを介して同様にメインのオシロスコープにて観測されメインの PC に記録される。サブのオシロスコープにより、テスト波とコレクターからのビーム電流、これら両者間の同期状態を監視することができる。アナライザーの分別グリッドへの任意電圧の印加は、メインの PC により GP-IB を介して電圧電源を制御することで行われる。RS-232C を介してメインの PC の命令を受けられるサブの PC は、AC モーター・ドライバーを制御することで、任意位置にプローブを移動させることができる。さらに、サブの PC はガス圧や放電電流などの間接的なプラズマ情報を知らせることができる。一連の実験データは、LAN にてメインの PC よりワークステーションへ転送され、画像化ソフトおよび解析ソフトにて視覚化およびグラフ化される。

なお、主な実験パラメータは Table2.1 に示す通りである。

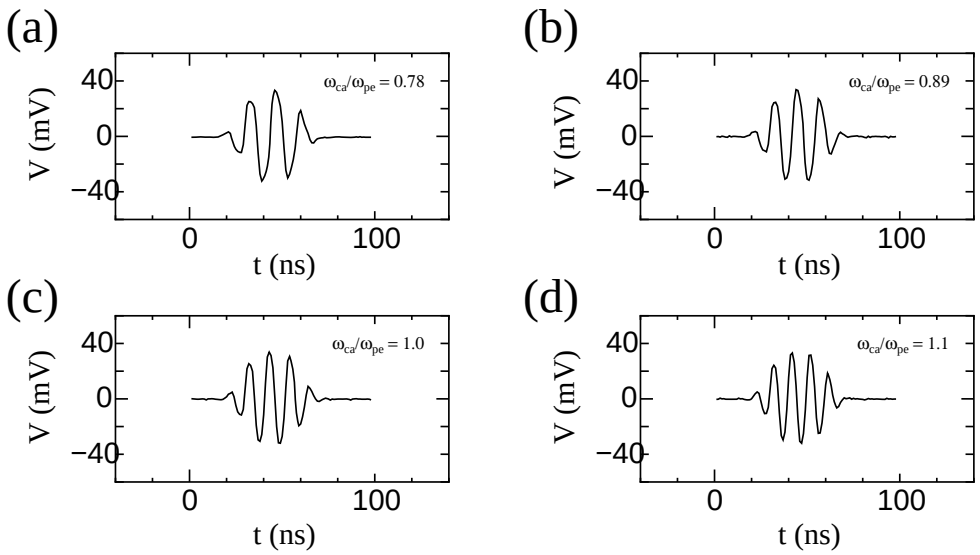


Fig. 2.7: Test waves; (a) $\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.78$, (b) $\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.89$, (c) $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$, and (d) $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.1$.

Table 2.1: Experimental parameters.

Argon gas pressure	$2.8 \times 10^{-5}\mathrm{Torr}$
Axial magnetic field	$0.01\mathrm{T}$
Electron-plasma temperature	$0.76\mathrm{eV}$
Electron-plasma density	$n_{\rm e} \simeq 1.0 \times 10^{14} 1/\mathrm{m}^3$
Electron-beam density	$n_{\rm b}/n_{\rm e} \simeq 0.15 - 1.2\%$
Electron-beam velocity	$v_{\rm b} \simeq 4.2 \times 10^6 \mathrm{m/s}$
Electron-plasma thermal velocity	$v_{\rm T}/v_{\rm b} \simeq 0.12$
Initial electron-beam spread	$\Delta v/v_{\rm b} \lesssim 0.02$
Electron-beam duration time	$\omega_{\rm pe} t \simeq 2.0 \times 10^3$
Carrier frequency of test wave	$\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.78 - 1.1$
Special wavenumber	$k_0 \equiv \omega_{\rm pe}/v_{\rm b} \simeq 1.3 \times 10^2 m^{-1}$
Electron-plasma frequency	$\omega_{\rm pe} \simeq 5.6 \times 10^8 s^{-1}$
Electron-cyclotron frequency	$3.1\omega_{\rm pe}$

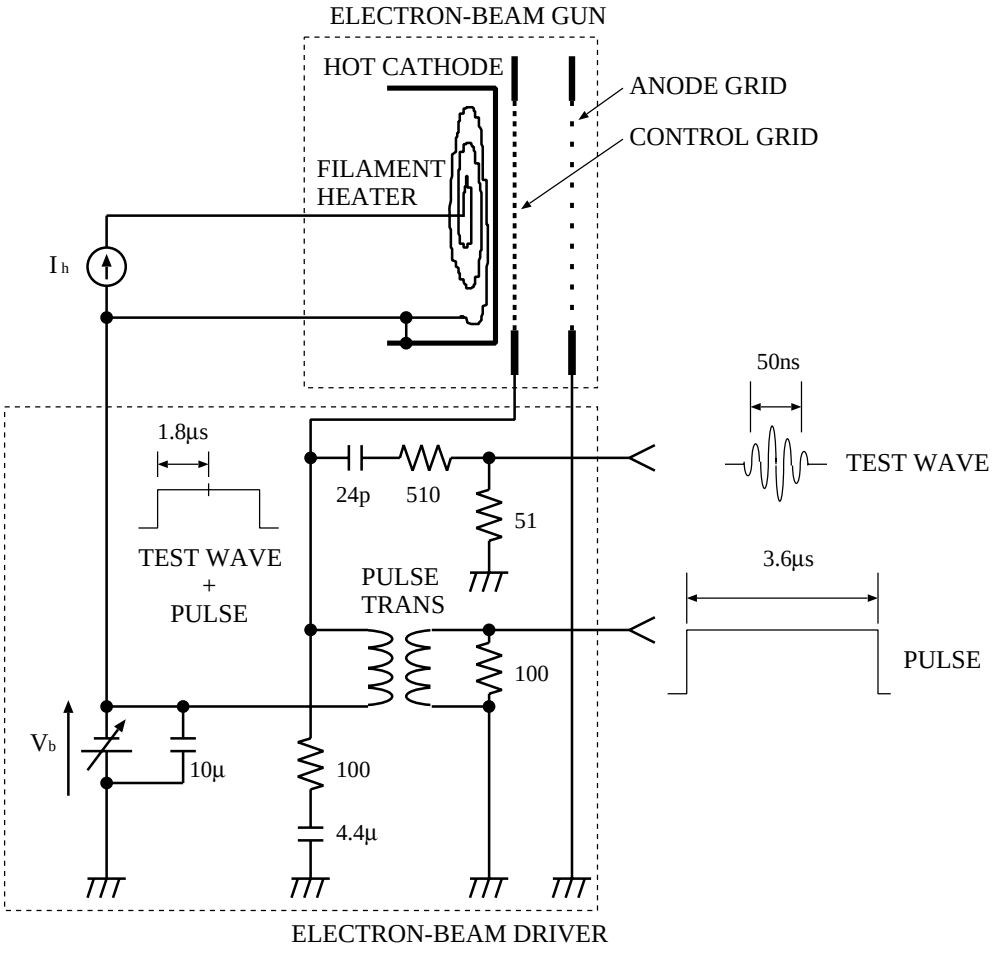


Fig. 2.8: Components of electron-beam gun and electron-beam driver.

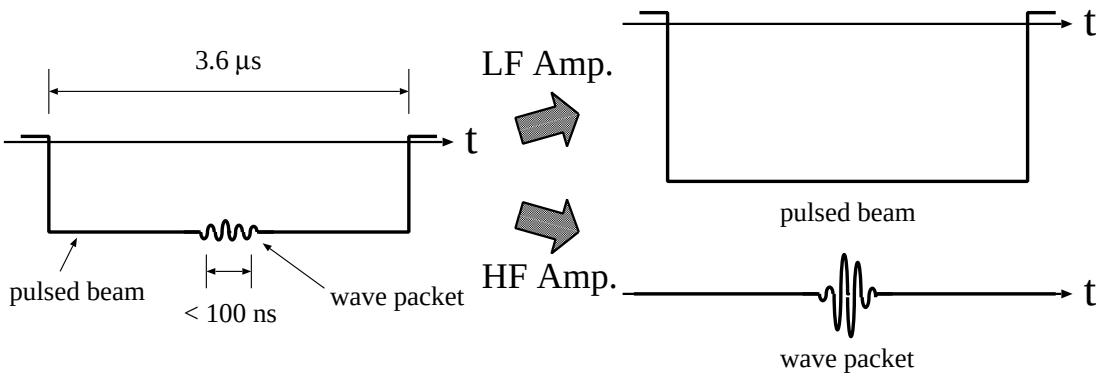


Fig. 2.9: Probe signal separation.

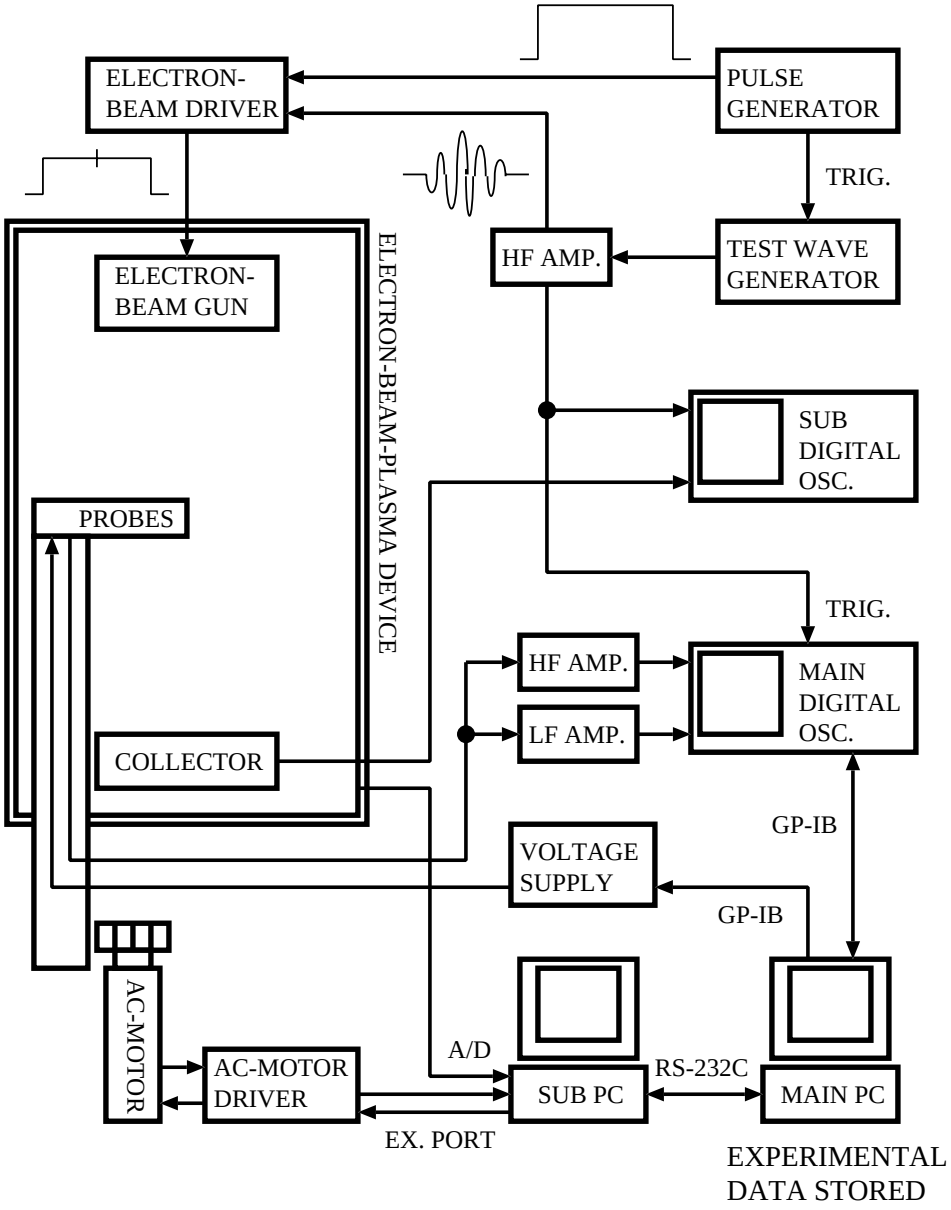


Fig. 2.10: Block diagram of experimental observation system.

Chapter 3

実験結果

3.1 波束の発展

ビーム密度が $n_{\rm b}/n_{\rm e} \simeq 0.15 - 1.2\%$、テスト波の搬送波周波数が $\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.78 - 1.1$ の場合において、位相速度 $v_\phi/v_{\rm b}$ を持つ波束の発展する様子を Table3.1 で列挙している番号の図で示す。(a) 図は z 空間における波束の振幅を $\omega_{\rm pe}t = 2.2$ の時間刻で表し、(b) 図は $k-\omega$ 平面における波束のパワー・スペクトル（以下、スペクトル）の強度を $\omega_{\rm pe}t = 17.9$ の時間刻でマップ表示（強度マップ）している。さらに、(c) 図は、強度マップにおいて最も大きい強度を示す $k-\omega$ 領域（以下、ピーク点）を、分散曲線上にプロットしている。なお、波の位相速度が $v_{\rm b}$ と等しい場合、規格要素の性質から $\omega_{\rm pe}t = k_0 z$ および $\omega/\omega_{\rm pe} = k/k_0$ が成り立つことに注意する。概して、波束は上流から下流へ伝搬しながら成長、飽和、減衰へと発展している。各強度マップにおけるピーク点は、時間と共にその強度や位置を変化させているが、ほぼ、ビーム・モードの分散曲線の付近に存在している。

$n_{\rm b}/n_{\rm e} \simeq 0.15 - 1.2\%$ および $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$ 一定の場合、Fig.3.2–3.4 および Fig.3.7 での波束の成長過程についての比較から、ビーム密度の増加が成長率の増加をもたら

Table 3.1: Figure lists against $n_{\rm b}/n_{\rm e}$, $\omega_{\rm ca}/\omega_{\rm pe}$ and $v_\phi/v_{\rm b}$.

Fig.	$n_{\rm b}/n_{\rm e}(\%)$	$\omega_{\rm ca}/\omega_{\rm pe}$	$v_\phi/v_{\rm b}$
3.2	0.15	1.0	0.89
3.3	0.3	1.0	0.89
3.4	0.6	1.0	0.84
3.5	1.2	0.78	0.86-0.96
3.6	1.2	0.89	0.85-0.90
3.7	1.2	1.0	0.84-0.88
3.8	1.2	1.1	0.82-0.86

している様子が分かる。波束の時間成長率のビーム密度依存性を Fig.3.9 に示す。両対数グラフ上において、点プロット（●印）が 1/3 乗曲線近くに並んでおり、成長過程における波束は、線形的に発展すると言える。$n_{\rm b}/n_{\rm e} \gtrsim 0.6\%$ の飽和過程において、波形に非対称的な歪みが現れている。例えば、Fig.3.3(a) における波束は、常に対称性が保たれているが、Fig.3.7(a) における飽和過程の波束は、ピーク振幅が $k_0 z \simeq 35-40$ 付近で局在するようにして非対称性が生じている。さらに、$\omega_{\rm pe} t \gtrsim 90$ のピーク振幅の位置より上流側では、下流側より波数が大きい（波長が短い）状態にあり、Fig.3.3(b) の $\omega_{\rm pe} t = 74-109$ および $92-127$ で、$k/k_0 = 1-1.5$ と $\omega/\omega_{\rm pe} \simeq 0.85-1.2$ の範囲で、波数バンドの拡がりとして現れている。このような歪みには、粒子捕捉などの非線形効果がもたらされている可能性がある。Fig.3.1 で示すように、局所的に減速した波は、波数に違いをもたらし、さらに、群速度の波数依存性（分散効果）のため波形の歪みを強める。減衰過程においては、波束の非対称性が弱められる様である。

$n_{\rm b}/n_{\rm e} \simeq 1.2\%$ 一定および $\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.78-1.1$ の場合、Fig.3.5–3.8 での波束の発展過程における比較から、搬送波周波数の増加が波形の非対称性を強めている事が分かる。上記の $\omega_{\rm ca}/\omega_{\rm pe}$ 一定の場合と同様に、局所的に波数に違いが生じており、強度マップからもその様子が確認できる。波束の時間成長率と強度マップのピーク点を、ビーム・プラズマ波の分散曲線と共に Fig.3.10 に示す。成長過程にある波束の特性を表している (a) 図の点プロット（●印）は、多少下方へ位置しているが、ほぼビーム・モードの分散曲線に沿っている。(b)-(c) 図および (d) 図の点プロット（●印）は、それぞれ、飽和過程および減衰過程にある波束の特性を表している。(a)–(d) 図の比較より、点プロットの分布が、$k/k_0 \simeq 1$、$\omega/\omega_{\rm pe} \simeq 0.94$ 付近へ移動していることが分かる。(d) 図における全ての点プロットの周波数が共通していることから、減衰過程にある波束においては、別のモードを示しているのかもしれない。

全般的に、変調不安定性による自己収束現象や複数の波束の放出は観測されなかった。

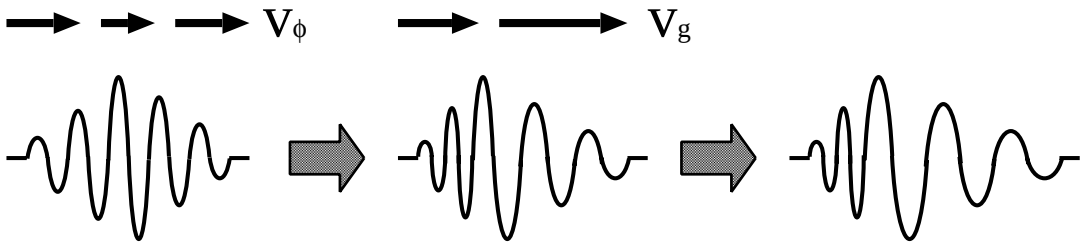


Fig. 3.1: Deformed wave packet.

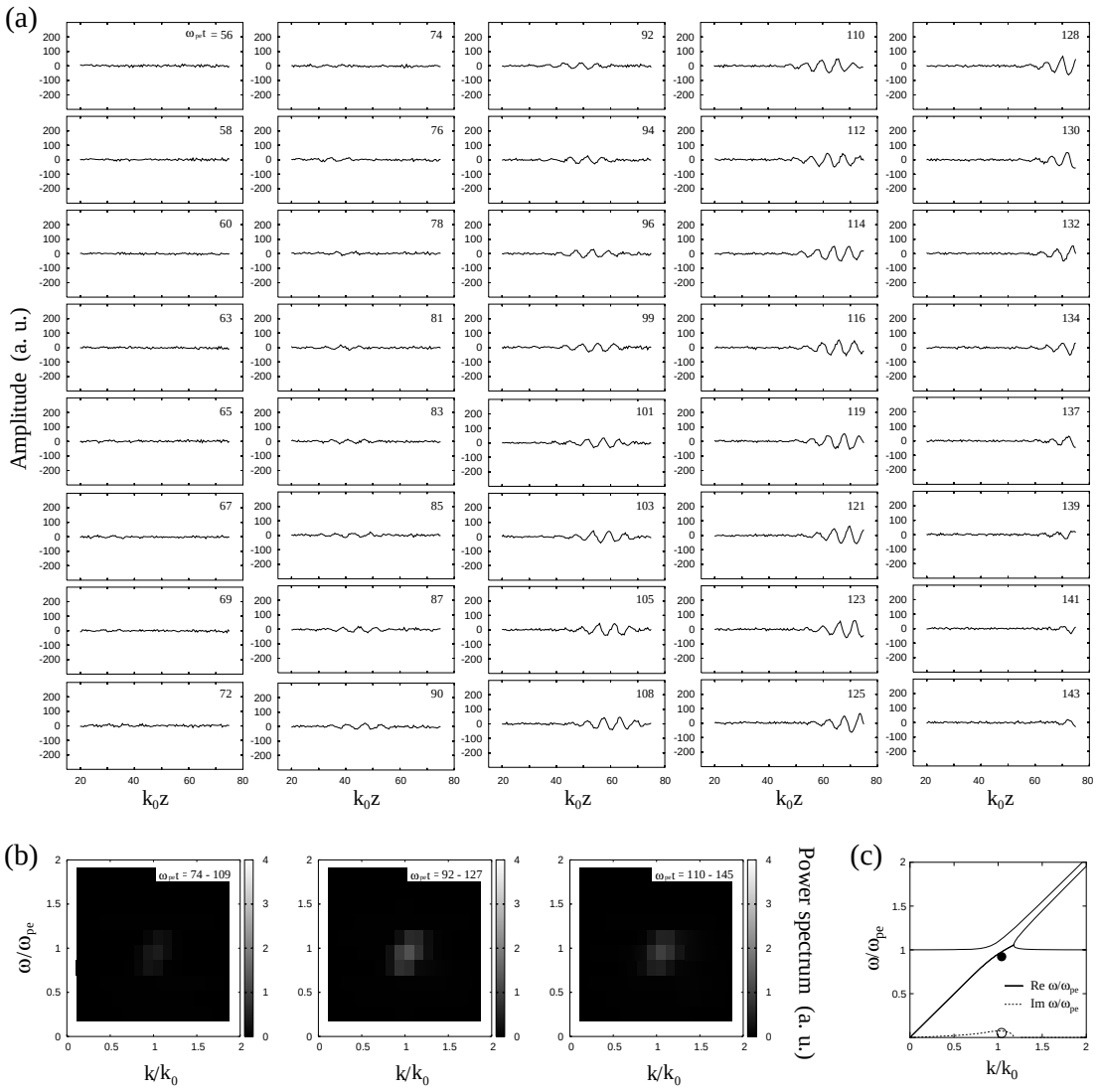


Fig. 3.2: Temporal evolution of wave packet for $n_{\rm b}/n_{\rm e} \simeq 0.15\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$; (a) amplitude vs $k_0 z$, (b) power spectrum vs k/k_0 and $\omega/\omega_{\rm pe}$, and (c) peak points of (b) with dispersion relation.

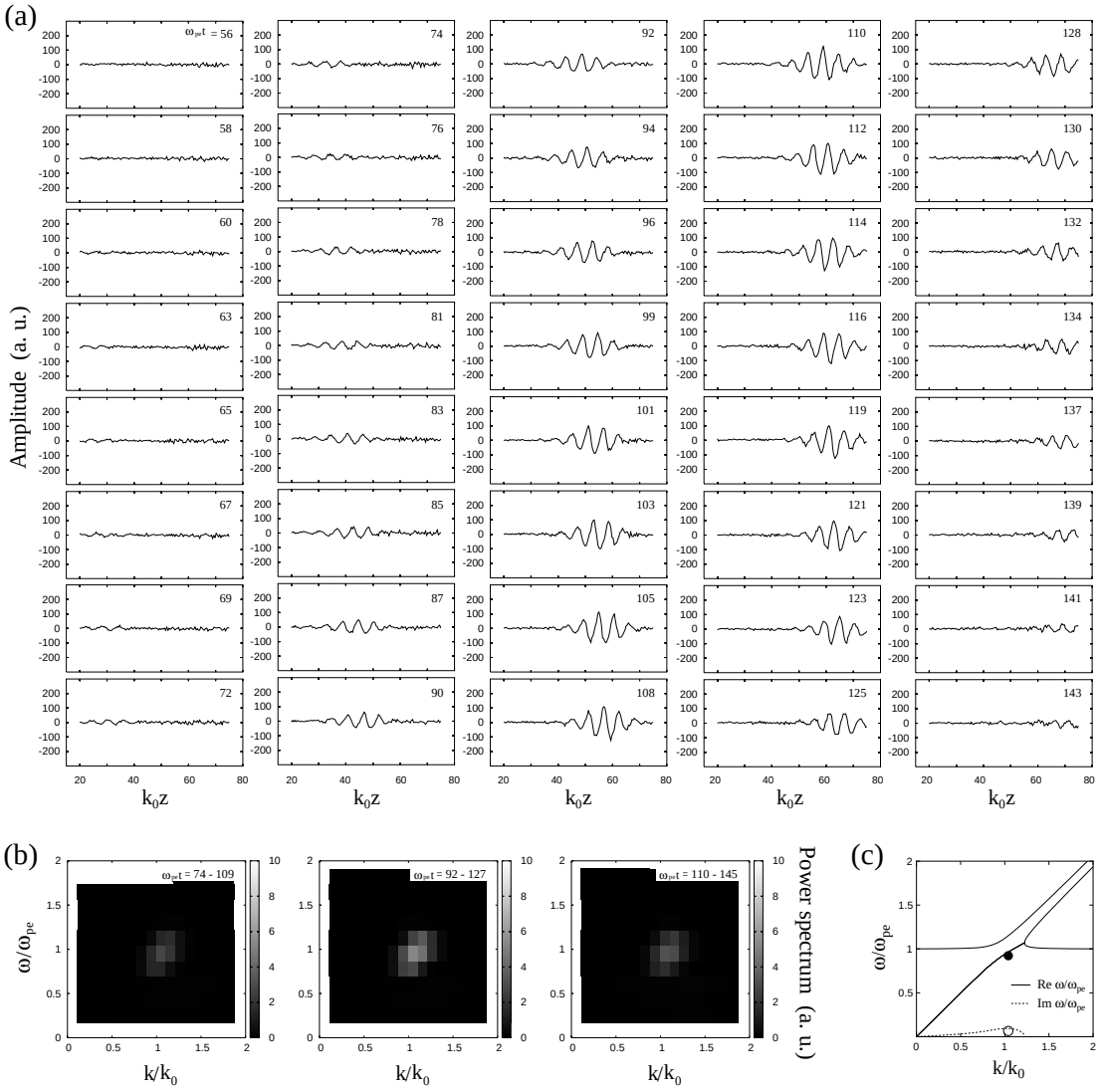


Fig. 3.3: Temporal evolution of wave packet for $n_{\rm b}/n_{\rm e} \simeq 0.3\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$; (a) amplitude vs $k_0 z$, (b) power spectrum vs k/k_0 and $\omega/\omega_{\rm pe}$, and (c) peak points of (b) with dispersion relation.

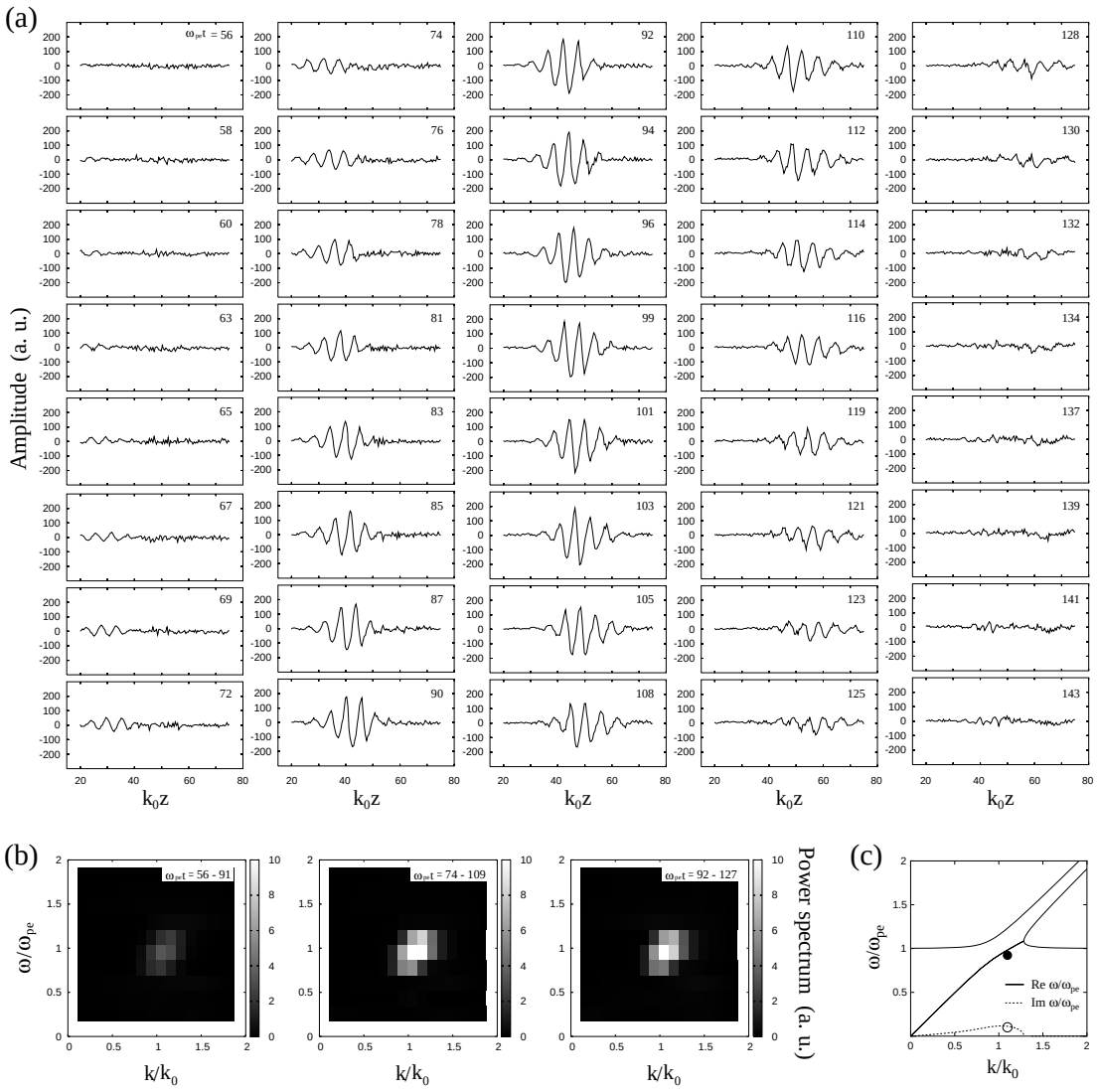


Fig. 3.4: Temporal evolution of wave packet for $n_{\rm b}/n_{\rm e} \simeq 0.6\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$; (a) amplitude vs $k_0 z$, (b) power spectrum vs k/k_0 and $\omega/\omega_{\rm pe}$, and (c) peak points of (b) with dispersion relation.

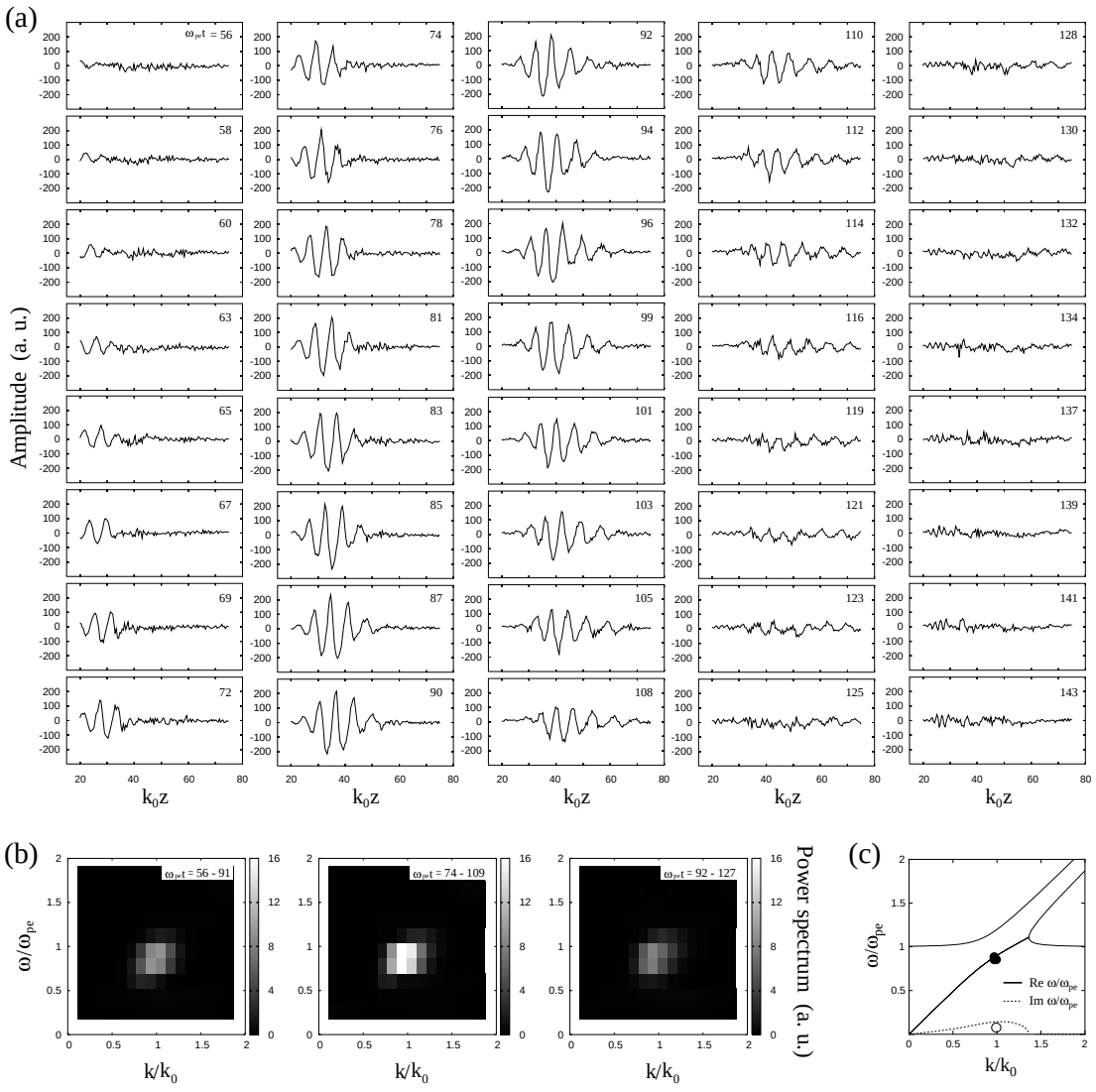


Fig. 3.5: Temporal evolution of wave packet for $n_{\rm b}/n_{\rm e} \simeq 1.2\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.78$; (a) amplitude vs $k_0 z$, (b) power spectrum vs k/k_0 and $\omega/\omega_{\rm pe}$, and (c) peak points of (b) with dispersion relation.

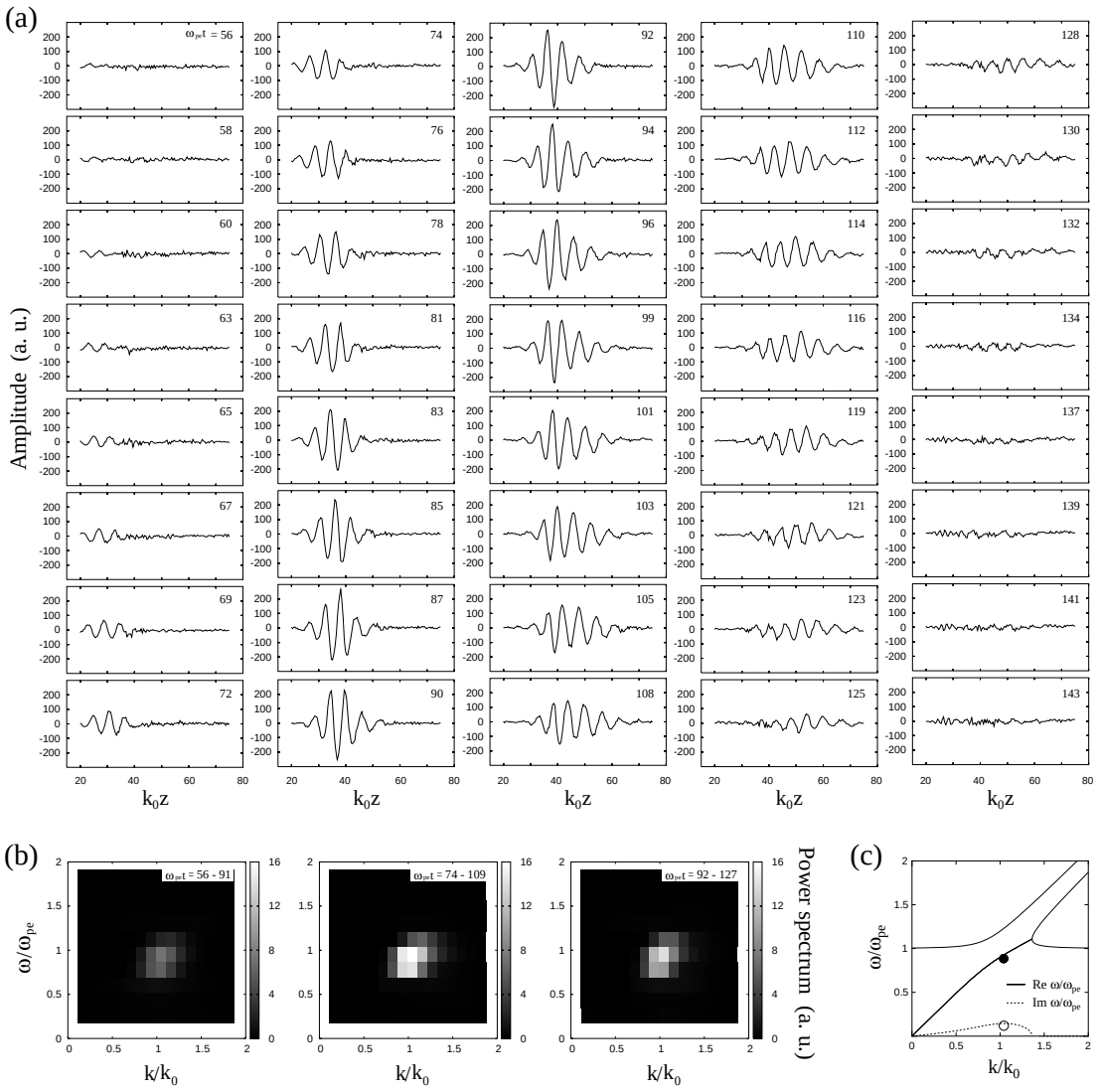


Fig. 3.6: Temporal evolution of wave packet for $n_{\rm b}/n_{\rm e} = 1.2\%$ and $\omega_{\rm ca}/\omega_{\rm pe} = 0.89$; (a) amplitude vs $k_0 z$, (b) power spectrum vs k/k_0 and $\omega/\omega_{\rm pe}$, and (c) peak points of (b) with dispersion relation.

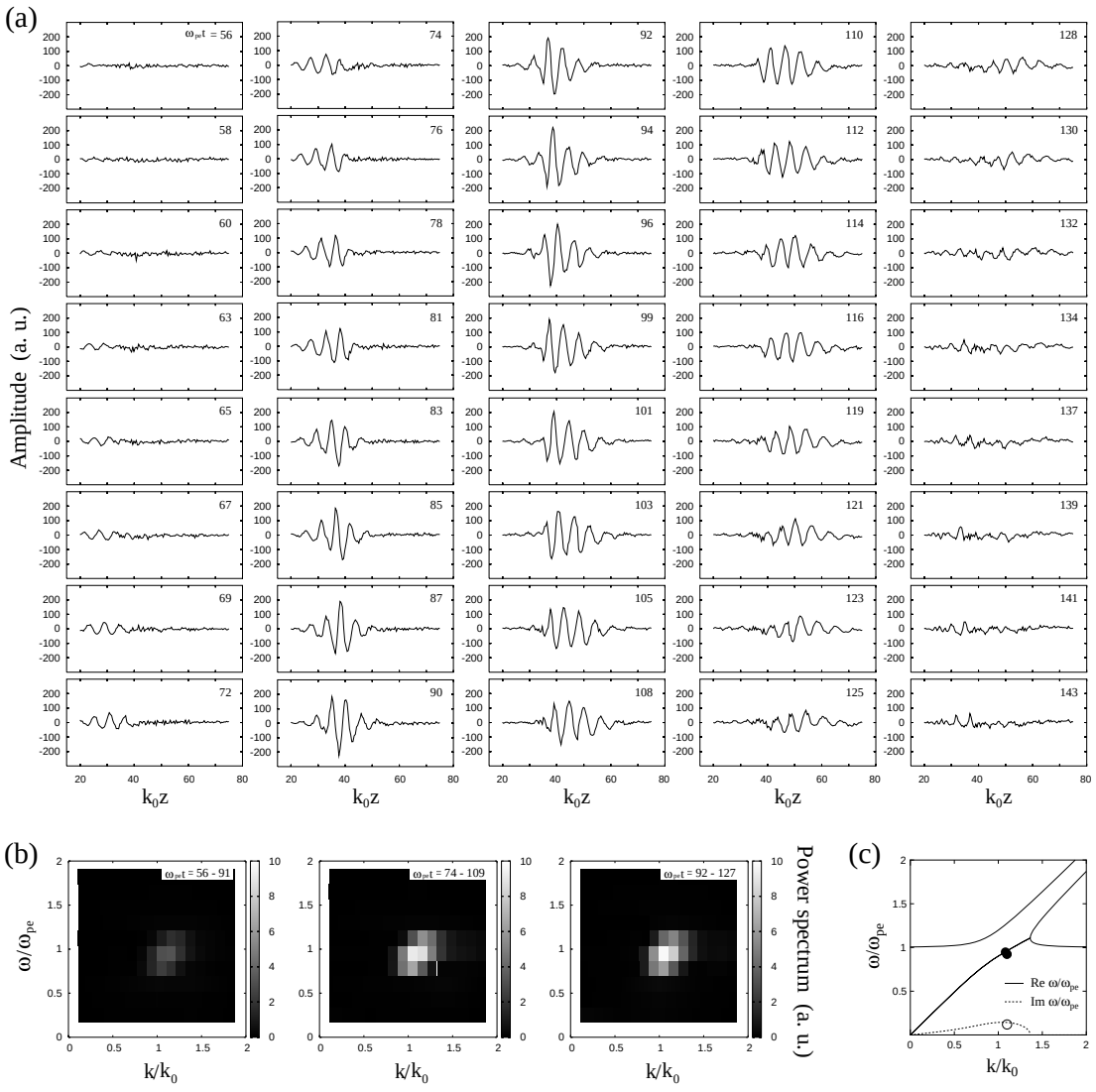


Fig. 3.7: Temporal evolution of a wave packet for $n_{\rm b}/n_{\rm e} \simeq 1.2\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$; (a) amplitude vs $k_0 z$, (b) power spectrum vs k/k_0 and $\omega/\omega_{\rm pe}$, and (c) peak points of (b) with dispersion relation.

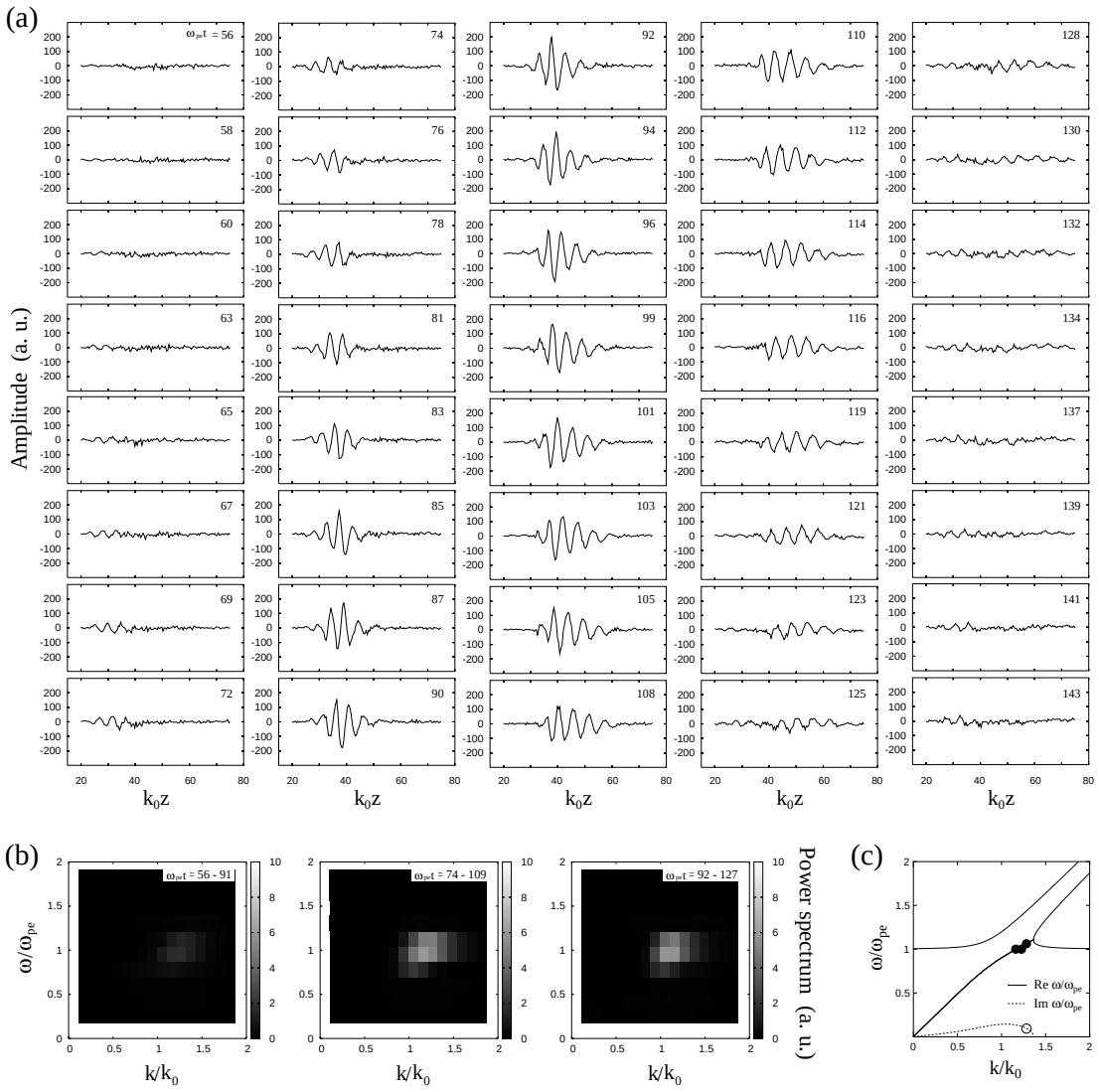


Fig. 3.8: Temporal evolution of wave packet for $n_{\rm b}/n_{\rm e} \simeq 1.2\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.1$; (a) amplitude vs $k_0 z$, (b) power spectrum vs k/k_0 and $\omega/\omega_{\rm pe}$, and (c) peak points of (b) with dispersion relation.

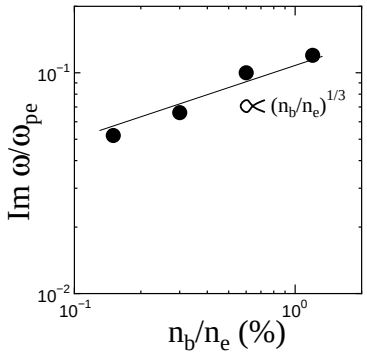


Fig. 3.9: Temporal growth rates $\mathrm{Im}\omega/\omega_{\mathrm{pe}}$ vs $n_{\mathrm{b}}/n_{\mathrm{e}}$ in the case of $\omega_{\mathrm{ca}}/\omega_{\mathrm{pe}} \simeq 1.0$.

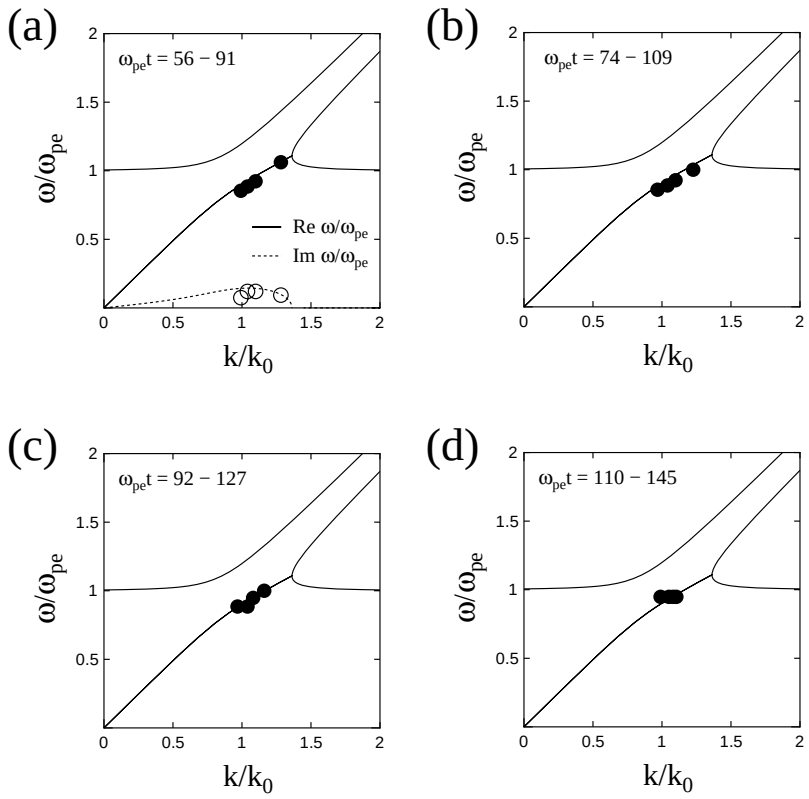


Fig. 3.10: Peak points in Fig. 3.5-3.8 with dispersion relations in the case of $n_{\mathrm{b}}/n_{\mathrm{e}} \simeq 1.2\%$; (a) on the growing stage, (b) and (c) on the saturating stage, and (d) on the damping stage.

3.2 電子ビームの速度分布関数

$n_{\rm b}/n_{\rm e} \simeq 0.3, 1.2\%$、$\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$ の場合について、低周波アンプを介し、$\omega_{\rm pe}t \simeq 60-140$ で時間平均して得られたビームの速度分布関数（以下、分布）$f_{\rm b}$ を Fig.3.12、3.13 に示す。(a) 図は、ビームの分布を $k_0z - v/v_{\rm b}$ の位相平面上にマップ表示したものであり、(b) 図は、各位置において側面表示したものである。低周波アンプの応答時間より短時間で時間平均を取っているで、情報の損失が生じていないことに注意する。Fig.3.12 は、$n_{\rm b}/n_{\rm e} \simeq 0.3\%$ の場合のビームの分布を表している。$k_0z \lesssim 50$ で $-v/v_{\rm b}$ 方向に緩やかな広がりを持っているが、ほぼ冷たいビームの様子を示している。$k_0z \simeq 50-60$ で急激な広がりを示しはじめ、$k_0z \gtrsim 60$ で平らになっている。Fig.3.13 は、$n_{\rm b}/n_{\rm e} \simeq 1.2\%$ の場合のビームの分布を表している。$k_0z \lesssim 30$ で通常の速度分布を示しているが、$k_0z \simeq 30-45$ で図中の矢印で示すような二つのタイプの拡がりが存在している。タイプ 1 (Type-I) は、$v/v_{\rm b} \simeq 0.95-1.1$ の範囲で $+v/v_{\rm b}$ 方向への通常の拡がりを示している。タイプ 2 (Type-II) は、$v/v_{\rm b} \simeq 0.75-0.95$ の範囲で $-v/v_{\rm b}$ 方向と $+k_0z$ 方向へ伸びており、異常な拡がりを示している。$k_0z \gtrsim 45$ ではほとんど平らになっている。波の位相速度と同程度くらいの速度を持ち、なおかつ、波のポテンシャルより低いエネルギーを持つビーム電子は、ポテンシャル内に捕捉され、その後、波の減衰などにより脱捕捉される。このようなビーム電子に対する散乱機構は、ビームの分布に広がりをもたらす可能性がある。この系でランダムに発生する自然励起波として、Fig.3.14 に $n_{\rm b}/n_{\rm e} \simeq 1.2\%$ の場合における一例を示す。(a) 図から分かるように様々な形をした波束が長時間にわたって存在し、(b) 図が示すようなブロードな周波数帯域を持っている。

$n_{\rm b}/n_{\rm e} \simeq 0.3, 1.2\%$、$\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.89-1.1$ の場合について、高周波アンプを介して得られたビームの分布 $f_{\rm b}$ を $k_0z - v/v_{\rm b}$ の位相平面上にマップ表示し、その時間発展を Fig.3.15–3.18 に示す。なお、時間変化を追いやすい様に、ある窪みについては矢印で指している。Fig.3.15 は、$n_{\rm b}/n_{\rm e} \simeq 0.3\%$、$\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$ の場合におけるビームの分布の時間発展を表している。密度の揺らぎのような複数の窪みが、z 軸と並行に上流から下流へと伝搬している。このときの窪みの中心となる $v/v_{\rm b} = 0.96$ は、Table3.1 で示した位相速度より大きく、これらの窪みが渦を意味している可能性は低い。$\omega_{\rm pe}t \gtrsim 110$ で、電子の塊のようなものが現れているが、何らかの原因で電子が放出されているようである。Fig.3.16–3.18 は、$n_{\rm b}/n_{\rm e} \simeq 1.2\%$、$\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.89-1.1$ の場合におけるビームの分布の時間発展を表している。概して、$v/v_{\rm b} = 0.95$ 付近で現れたドーナッツ状の分布は、上流から下流へ、かつ、$-v/v_{\rm b}$ 方向へ移動している。この移動過程において、窪みは徐々に大きくなり、そこを取り巻く電子の集まりも大きくなっている。Fig.3.11 に、$n_{\rm b}/n_{\rm e} \simeq 1.2\%$、$\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$ の場合におけるビームの分布と波束の一例を示す。ドーナッツ状の分布の中心となる $v/v_{\rm b} = 0.88$ は、Table3.1 で示した位相速度とほぼ一致し、また、波の山の位置とも一致している。これらの状態は、Fig.1.3 で描かれるような理論的イメージとよく合っているので、先のドーナッツ状の分布は渦であると考

えられ、波束のポテンシャルに電子が捕捉されている状態を表している。波束の振幅が最大となる位置 $k_0 z$ で、渦が最大規模となっているので、捕捉電子により波束に歪みがもたらされている可能性がある。渦の移動の軌跡が、ほぼタイプ２の存在位置と一致することから、捕捉電子がタイプ２の形成に関っている可能性がある。

Fig.3.19 に、$\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.89 - 1.1$ の場合におけるドーナッツ状の速度半径と波束のポテンシャルとの関係を示す。このときのポテンシャルの振幅の大きさは、1.4 式から最大で約 0.3V と見積もられる。このプロットにおいて直線性が現れていることから、速度半径は捕捉半径を意味し、さらに、先で示されたドーナッツ状の構造が、渦であることを支持している。以上の結果から、この系で発生する Fig.3.14 のような自然励起波が、ビーム電子に対し捕捉や脱捕捉を導き、タイプ２のような散乱されたビームの分布を形成するものと予想される。電子捕捉により作り出される渦構造は、コンピューター・シミュレーションでも観測されている [18, 19]。残念ながら、本実験においては渦の回転方向は判別できないが、シミュレーション結果から、時計回りに回転していると予想される。$\omega_{\rm pe} t \lesssim 99$ において、$v/v_{\rm b} \simeq 0.95 - 1.1$ の範囲で高速度側に存在するノイズは、タイプ１の形成を意味しているのかもしれないが、明確なことは分からない。

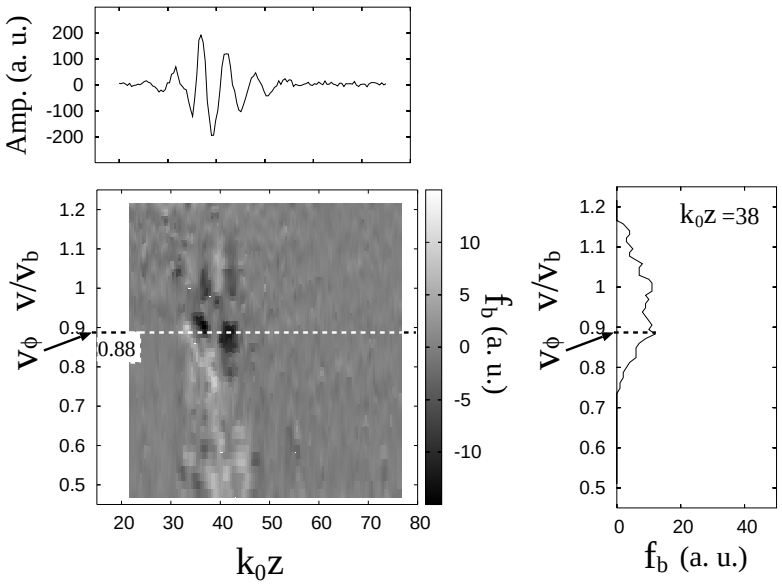


Fig. 3.11: Distribution function and wave packet in the case of $n_{\rm b}/n_{\rm e} \simeq 1.2\%$, $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$.

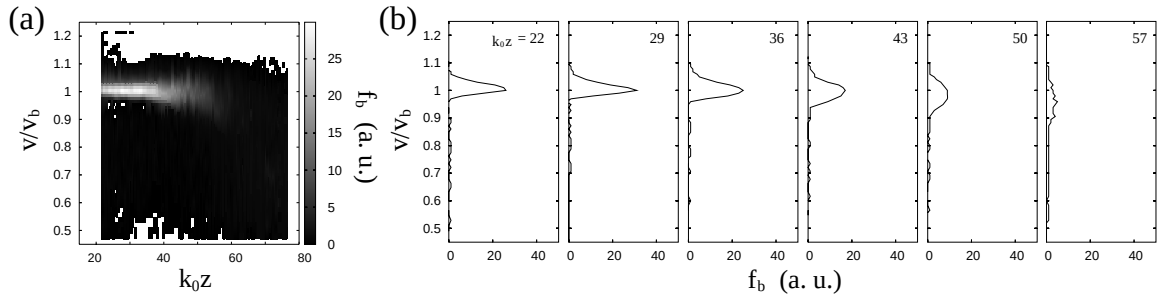


Fig. 3.12: Distribution function of electron-beam $f_{\rm b}$ vs $k_0 z$ and $v/v_{\rm b}$ for $n_{\rm b}/n_{\rm e} \simeq 0.3\%$ and $\omega_{\rm pe} t \simeq 60 - 140$; (a) from a top view, and (b) from side views.

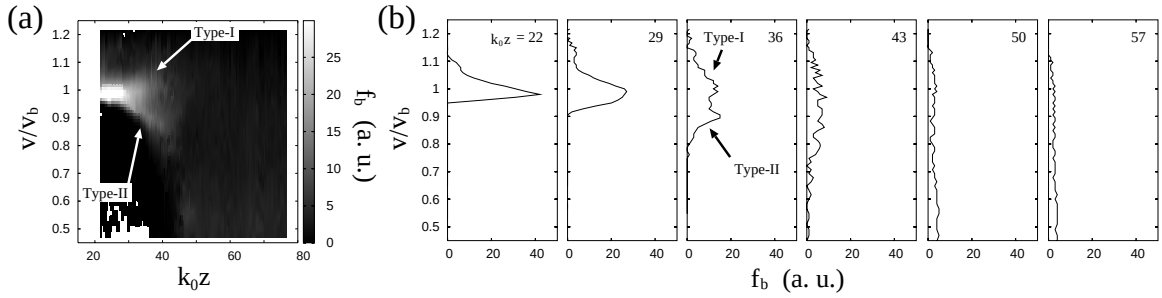


Fig. 3.13: Distribution function of electron-beam $f_{\rm b}$ vs $k_0 z$ and $v/v_{\rm b}$ for $n_{\rm b}/n_{\rm e} \simeq 1.2\%$ and $\omega_{\rm pe} t \simeq 60 - 140$; (a) from a top view, and (b) from side views.

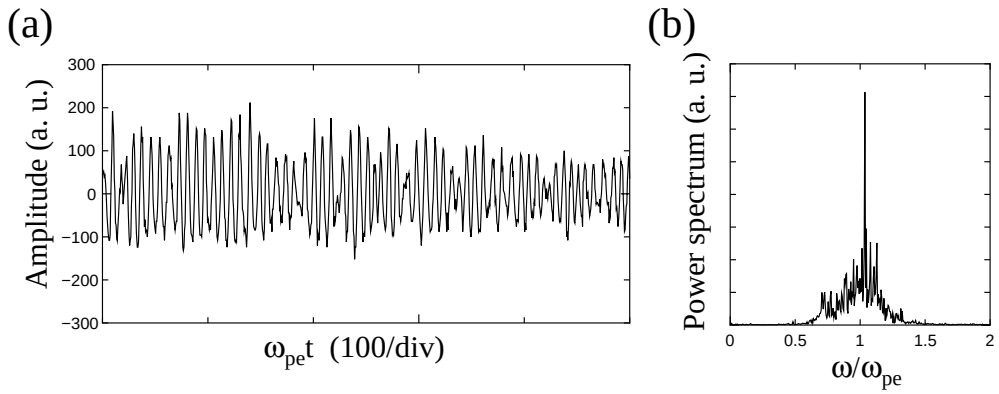


Fig. 3.14: Spontaneous waves for $n_{\rm b}/n_{\rm e} \simeq 1.2\%$; (a) amplitude, and (b) power spectrum.

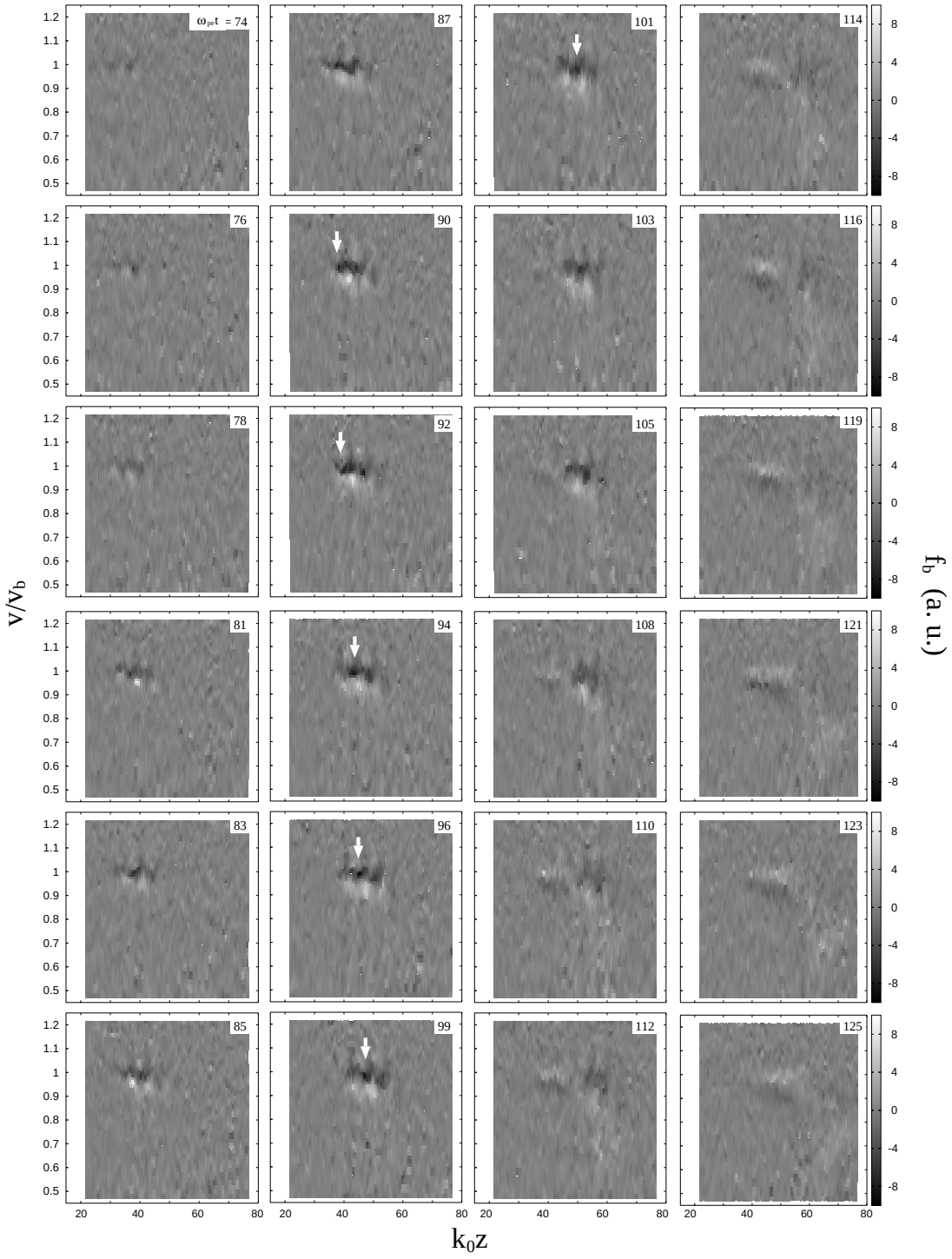


Fig. 3.15: Temporal evolution of distribution function of electron-beam $f_{\rm b}$ vs $k_0 z$ and $v/v_{\rm b}$ for $n_{\rm b}/n_{\rm e} \simeq 0.3\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$.

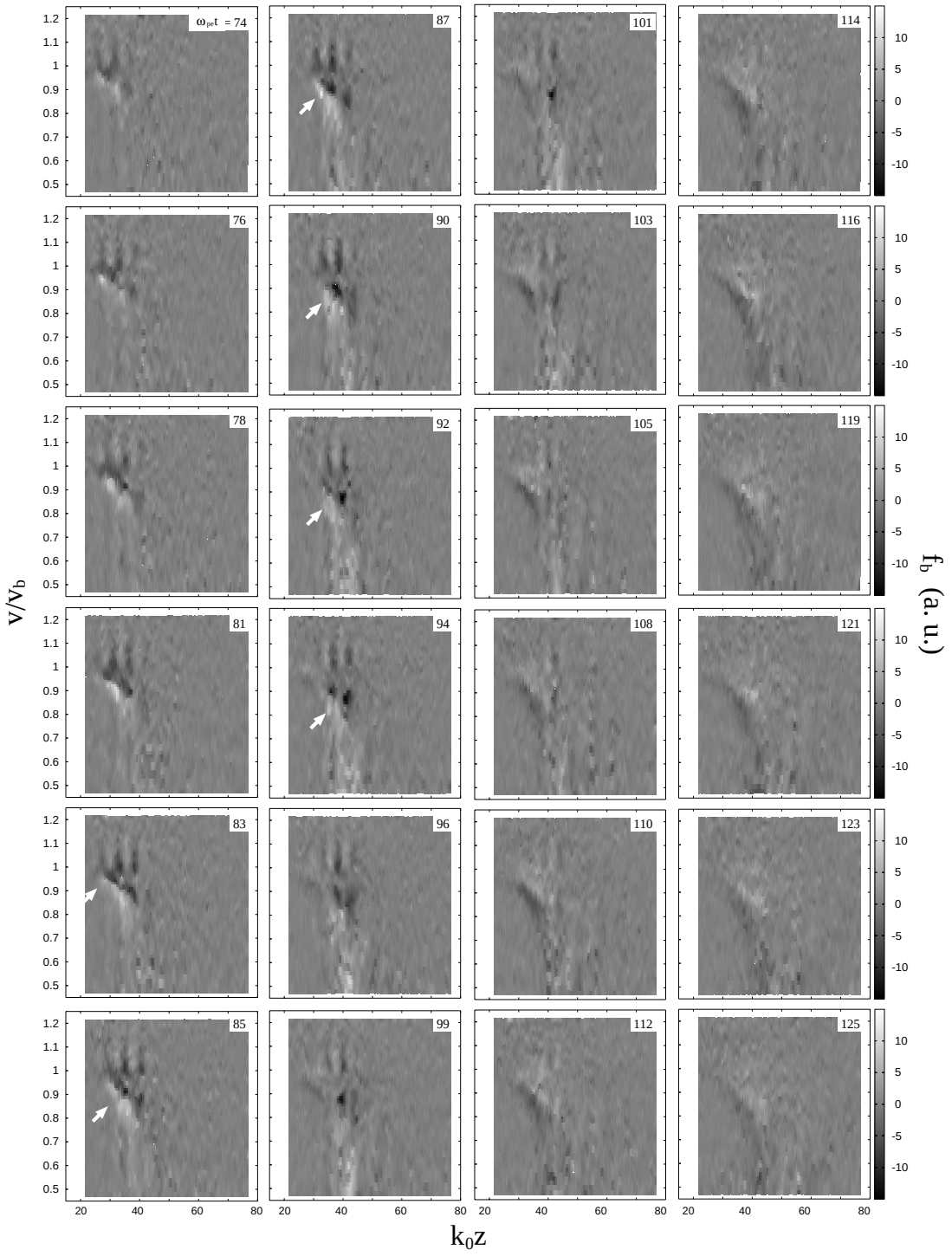


Fig. 3.16: Temporal evolution of distribution function of electron-beam $f_{\rm b}$ vs $k_0 z$ and $v/v_{\rm b}$ for $n_{\rm b}/n_{\rm e} \simeq 1.2\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.89$.

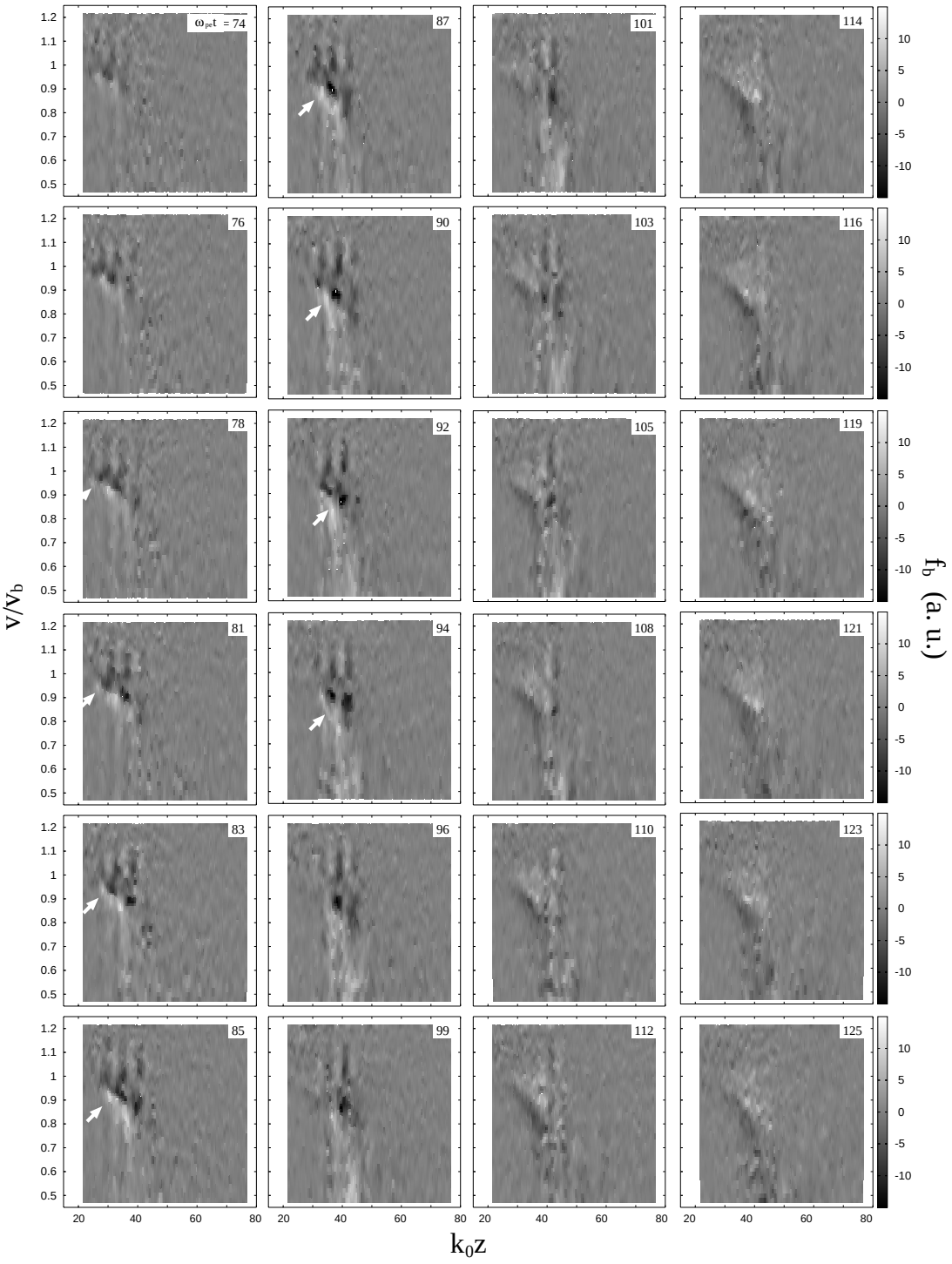


Fig. 3.17: Temporal evolution of distribution function of electron-beam $f_{\rm b}$ vs $k_0 z$ and $v/v_{\rm b}$ for $n_{\rm b}/n_{\rm e} \simeq 1.2\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$.

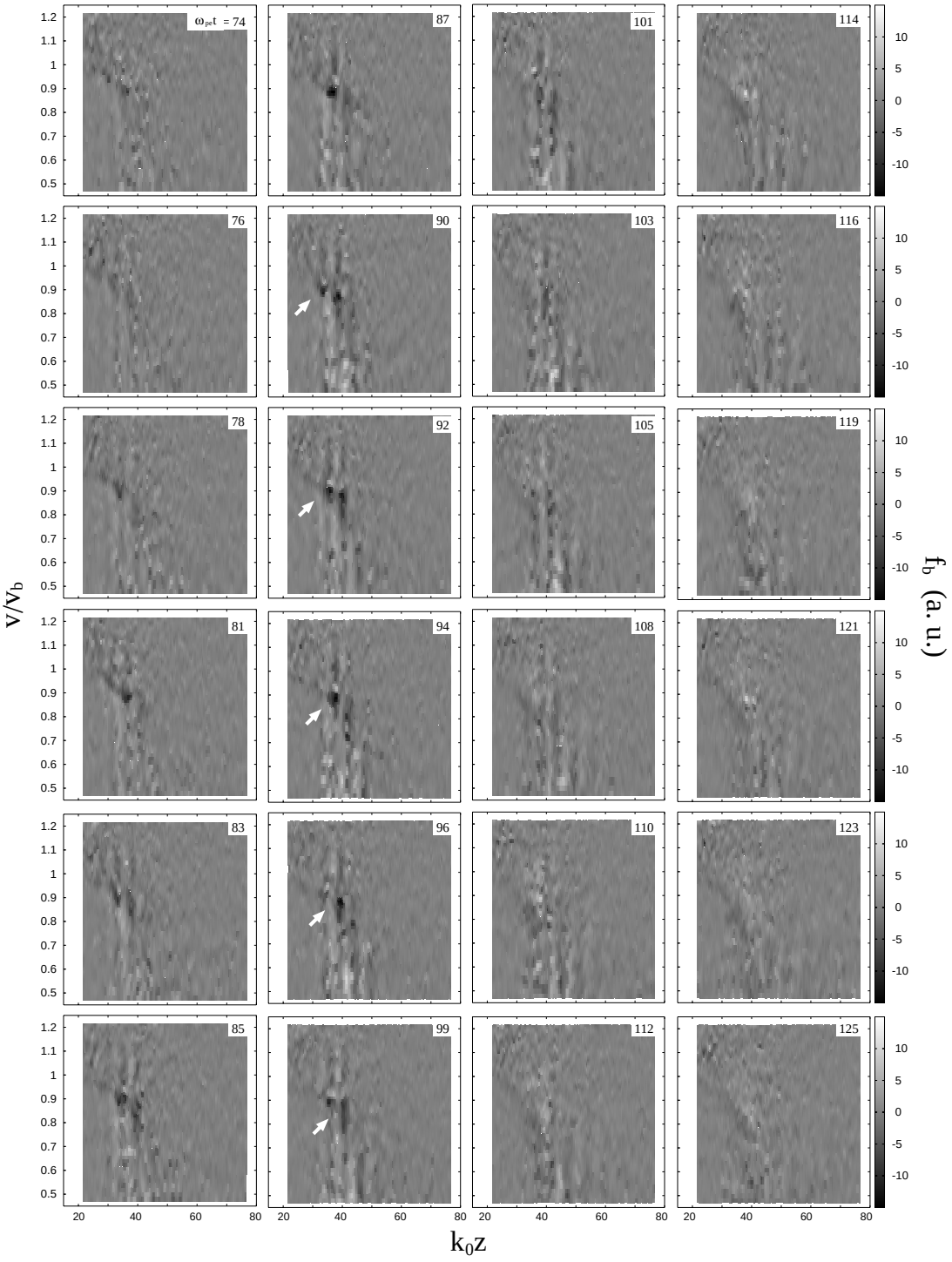


Fig. 3.18: Temporal evolution of distribution function of electron-beam $f_{\rm b}$ vs $k_0 z$ and $v/v_{\rm b}$ for $n_{\rm b}/n_{\rm e} \simeq 1.2\%$ and $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.1$.

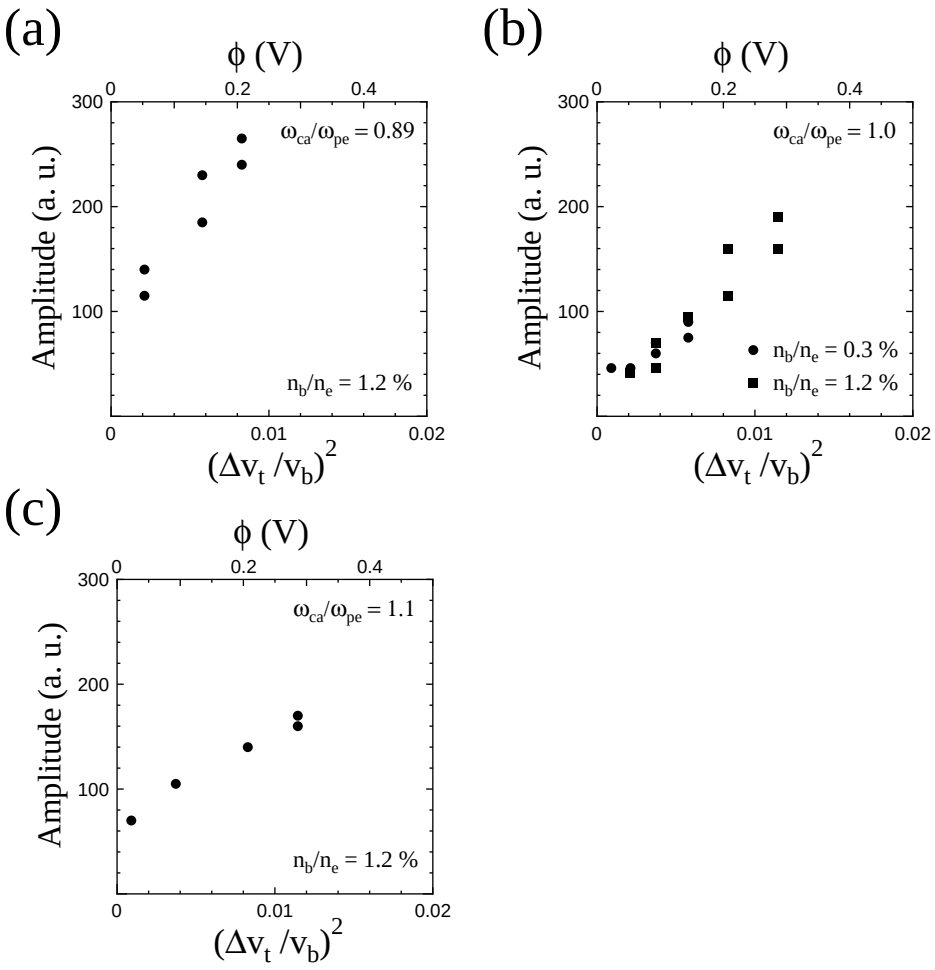


Fig. 3.19: Amplitude vs $(\Delta v_{\rm t}/v_{\rm b})^2$; (a) at $\omega_{\rm ca}/\omega_{\rm pe} \simeq 0.89$, (b) at $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.0$, and (c) at $\omega_{\rm ca}/\omega_{\rm pe} \simeq 1.1$.

Chapter 4

結論

電子ビーム・プラズマ系における非線形現象を探るべく、電子の応答時間スケールで空間発展する波束に対し、同軸プローブとエネルギー・アナライザーを用いて、ポテンシャルと電子ビームの速度分布関数の実験的観測を行った。同軸プローブによる観測から、波束についての成長、飽和、減衰からなる発展過程が示された。エネルギー・アナライザーによる観測から、プラズマ密度に対しビーム密度が 1.2% の場合、低周波帯域においては、高速側への広がり（タイプ１）と低速側へ伸びた異常な広がり（タイプ２）が示され、高周波帯域においては、時間発展をする渦に似たドーナッツ状の分布が示された。これらの観測結果から次のことが分かった。

（１）波束の位相速度とドーナッツ状の分布の中心の速度は一致する。
（２）ドーナッツ状の分布の速度半径の二乗がポテンシャルの大きさに比例していることから、この速度半径は捕捉半径を表している。
（３）（２）において、ビーム電子を捕捉するポテンシャルの大きさは、最大で約 0.3V と見積もられる。
（４）（１）－（３）より、ドーナッツ状の分布が、波束のポテンシャルにビーム電子が捕捉されて形成される渦であることを示している。
（５）以上より、この系で発生する自然励起波のポテンシャルが、ビーム電子に対し捕捉や脱捕捉などの散乱過程をもたらすことで、タイプ２のような分布が形成されると予想される。

今後の課題として、次のことが上げられる。

（６）渦の回転方向が判別できるように観測システムを再構築する必要がある。
（７）タイプ１の形成メカニズムを明らかにする。

Bibliography

[1] 日本物理学会, 「プラズマと核融合」(丸善, 1976)

[2] 田中基彦、西川恭治, 「高温プラズマの物理学」(丸善, 1996)

[3] 渡辺慎介, 「ソリトン物理入門」(培風館, 1995)

[4] R. J. Briggs, *Advances in Plasma Physics*, ed. A. Simon and W. B. Thompson (Interscience, New York, 1971) Vol. 4, p. 43.

[5] F. F. Chen, *Introduction to Controlled Fusion* (Plenum, New York, 1990)

[6] A. Hasegawa, *Plasma Instabilities and Nonlinear Effects* (Springer-Verlag, Berlin Heidelberg New York, 1975)

[7] M. V. Goldman, Rev. Mod. Phys. **56** 709 (1984).

[8] H. Ikezi, R. J. Taylor and D. B. Baker, Phys. Rev. Lett. **25** 11 (1970).

[9] G. A. Asker'yan, Sov. Phys. JETP **15** 1088 (1962).

[10] A. Y. Wong and P. Y. Cheung, Phys. Rev. Lett. **52** 1222 (1984).

[11] V. E. Zakharov, Sov. Phys. JETP **35** 908 (1972).

[12] T. Intrator, C. Chan, N. Hershkowitz and D. Diebold, Phys. Rev. Lett. **53** 1233 (1984).

[13] L. A. Ostrovskii, Sov. Phys. JETP **24** 797 (1967).

[14] K. Yamagiwa, T. Itoh and T. Nakayama, *Invited Papers, XXIIIrd ICPIG 1997, Toulouse* (EDP Sciences, 1997) Vol. 7, p. C4-413.

[15] Takeda T. and Yamagiwa K., J. Plasma Fusion Res. **79** 323 (2003).

[16] K. N. Leung, T. K, Samec and A. Lamm, Phys. Lett. **51A** 490 (1975).

[17] R. L. Stenzel, W, Gekelman, N. Wild, J. M. Urrutia and D. Whelan, Rev. Sci. Instrum. **54** 1302 (1983).

[18] S. Kainer, J. Dawson and R. Shanny, Phys. Fluids. **15** 493 (1972).

[19] K. Akimoto, Y. Omura and H. Matsumoto, Phys. Plasmas. **3** 2559 (1996).

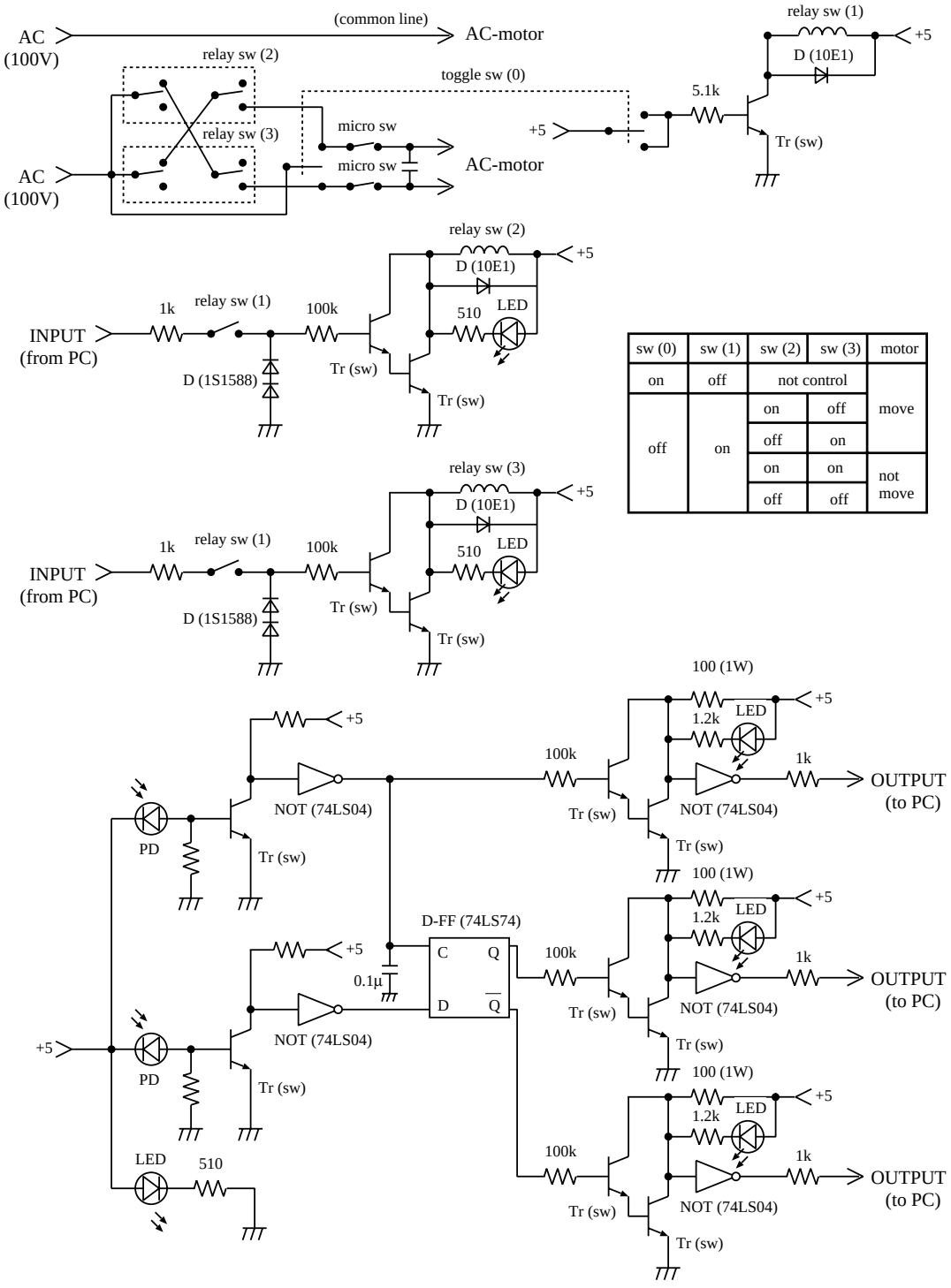
Appendix A

電子回路

●モーター・ドライバー (Fig.A.1)

●ＬＦアンプ (Fig.A.2)

●ラングミュア・プローブ測定用 (Fig.A.3)



sw (0)	sw (1)	sw (2)	sw (3)	motor
on	off	not control		move
off	on	on	off	
		off	on	
		on	on	not move
		off	off	

Fig. A.1: Motor driver.

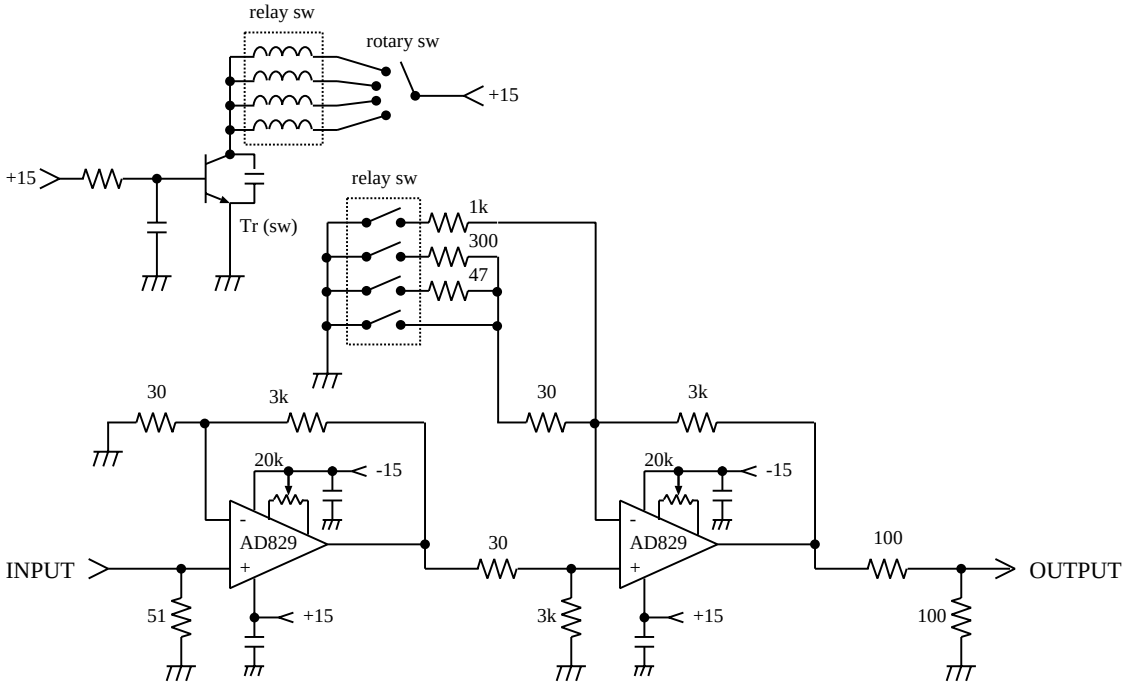


Fig. A.2: LF amplifier.

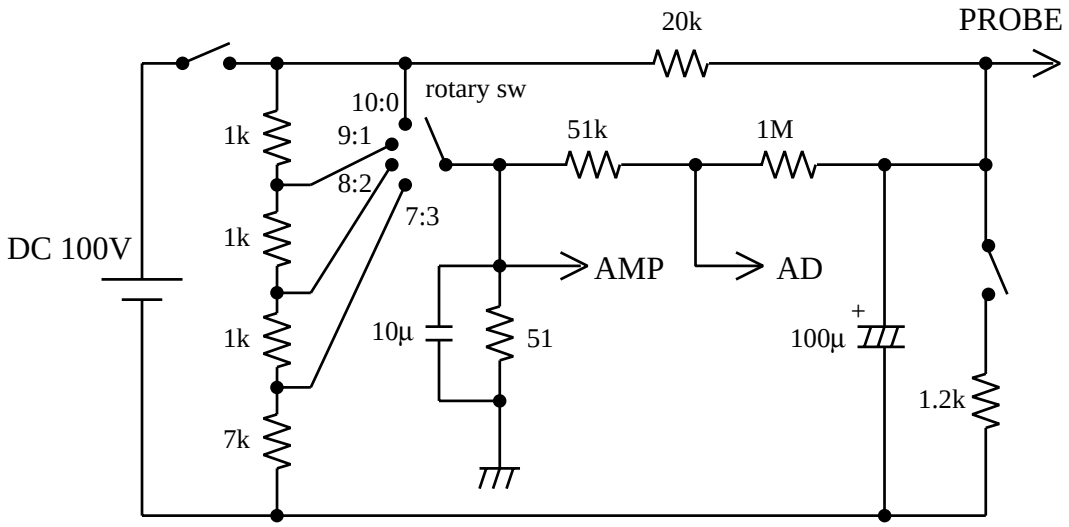


Fig. A.3: Langmure probe measurement.

Appendix B

プログラム

●測定用 (n88basic for pc98 ms-dos)
メイン PC ——- hpa.bat, hp2a.bat, hp.ini [1–27]
サブ PC ———- mon.bat [27–31]
ラングミュア — lan.bat, lan.ini [31–49]

●解析用 (g++ for unix)
UNIX 形式変換 — d2u.c [49–51]
DATA 統合 ——— tz3d.c, zv3d.c [51–66]
3D-FFT ————- fft3d.c [66–78]
平滑処理 ———— square-average.c, time-average.c [78–90]
分散関係 ———— dispersion.c, parameter.ini [90–94]

```
1000 '  HPA.BAS (n88BASIC)    ''''''''''''''''''''''''  Since Jan. 2000  '
1010 '                                                                    '
1020 '  Automatical Measurement Softwear                                  '
1025 '  for HP545 Series with PC98 and GP-IB Board                        '
1030 '                                                                    '
1040 '                                    Programed by TAKEDA Tsuyoshi    '
1050 '                                                                    '
1060 '                                                                    '
1070 '  References: NEC,                                                  '
1080 '              PC-8897 GP-IB Interface Board USER'S MANUAL (1990)    '
1085 '              PC-9801-29N GP-IB Interface Board USER'S MANUAL       '
1090 '                                                                    '
1100 '              K. YAMAGIWA and M. SASAKI,                            '
1110 '              Rep. Fact. Sci. SHIZUOKA UNIV. Vol.26, 31 (1992)      '
1120 '                                                                    '
1130 '                                                                    '
1140 '  Initial file, "HP.INI".                                           '
1160 '  #Initial File for HP*.BAS                                         '
1170 '  #                                                                 '
1180 '  OSC,ADD,CH,RMS,SAV,WAN                                            '
1190 '  22A, 2,  1,  Y,  Y,  Y :ADD -- Address of OSC (1 - 30)            '
1200 '  22A, 2,  2,  Y,  Y,  Y :CH  -- Channel of OSC (1 - 4)             '
1210 '  10A, 3,  1,  Y,  Y,  Y :RMS -- Root Mean Square (Y or N(average)) '
1220 '  10A, 3,  2,  Y,  Y,  Y :SAV -- Save (Y or N)                      '
1225 '  10A, 3,  2,  Y,  Y,  Y :WAN -- Warn (Y or N)                      '
1230 '  #                                                                 '
1240 '  PC98-ADD, 1     :Address of GPIB-board.                           '
1250 '  RPT-NMB,  2     :Repeated number as the same condition.           '
1260 '  WRN-LMT,  32    :Warned points on scaling from bottom and top.    '
1270 '  H-RJCT,   0     :Rejected points on RMS from left and right.      '
1280 '  SAVE-DIR, B:¥   :Saved directory.                                 '
1285 '  WT-TIME,  7     :Waitting time (sec).                             '
1290 '                                                                    '
1300 '  Select screen, "SELECT![2/4/6/8/0/./+/*/-/y/n]".                  '
1301 '  2/4  :Back step on data number.                                   '
1302 '  6/8  :Forward step on data number.                                '
1303 '  0    :Getting and showing data.                                   '
1304 '  .    :Saving data.                                                '
1305 '  +    :(Next step and) getting and saving data.                    '
1306 '  *    :(Next step and) getting, showing and saving data.           '
1307 '  -    :Showing spectram by FFT.                                    '
1308 '  y    :"Yes" or next.                                              '
1309 '  n    :"No" or exit.                                               '
1310 '                                                                    '
1312 '  Pressing HELP key, change to Full Automation Mode or Normal Mode. '
1314 '                                                                    '
1316 '                                                                    '
1320 '  NOTICE!: This program is available for the following conditions:  '
```

```
1330 '           Sampling time rate,      1 (ns/points)                  '
1340 '           Max vertical points,     256  or 1024                   '
1350 '           Max horizontal points,   8000 or 500                    '
1360 '                                                                   '
1362 '           Need another controllable PC connected with RS232C.     '
1365 '                                                                   '
1370 ''''''''''''''''''''''''''''''''''''''''  Last modified in Apr. 2003  '
1380 '
1390 '
1400 ''''''''''''''''''''''''''    MAIN  ROUTINE    ''''''''''''''''''''''''''
1410 '
1420      *MAIN
1430 '
1440 ON ERROR GOTO *ERRORS
1450 CMD DELIM=0: CMD TIMEOUT=5
1460 ISET IFC :ISET REN
1470 DEFINT A-Z
1480 '
1490 GOSUB *DEFINES :GOSUB *INIT
1510 GOSUB *RS232C
1520 K=0 :L=Z0+1000 :LL=0 :DL=DZ :M0=1 :M=M0 :MN=NMB :DM=1
1530 WHILE 1
1580 GOSUB *FAQ
1590 IF R=1 THEN GOSUB *FIN               'Finish Program when R=1
1595 IF AT=1 AND L=ZN+1000 AND M>MN THEN GOSUB *QUIT
1635 WEND
1640 '
1650      *FIN
1660 '
1670 'GOSUB *RMSSAVE
1680 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
1690 PRINT "FINISH ?" :COLOR@ (0,0)-(31,0),6 :BEEP
1700 GOSUB *FAQ
1710 IF R=0 THEN GOSUB *QUIT
1730 IF R=1 THEN R=0 :RETURN
1750 '
1760      *QUIT
1770 '
1780 LOCATE 0,1 :PRINT SPACE$(32) :LOCATE 0,1
1790 PRINT "FINISHED !!" :COLOR@ (0,1)-(31,1),2
1800 BEEP :BEEP :BEEP :BEEP :BEEP
1810 CLOSE :CLEAR :CLS 3 :CONSOLE 0,25,1,1
1820 KEY OFF :STOP OFF
1825 COM OFF :CLOSE #2
1830 ON ERROR GOTO 0
1840 IRESET REN
1850 END
1860 '
```

```
1870 ''''''''''''''''''''''''''    SUB  ROUTINE    ''''''''''''''''''''''''''
1880 '
1890      *DEFINES
1900 '
1902 MVR=4                    'MVR: Min. V-Range (mv/div)
1905 HN=4000                  'HN: Receiving Time to another PC
1907 AT=0                     'When AT=1, automation mode runs
1910 DIM WD(3,8000), AD(4), RDO!(4,10)  ', RD!(4,450)
1920 DIM X(1024), XX(1024), MIN(4) ,MAX(4), WN(2)
1930 DIM FDR#(1024), FDI#(1024), FD#(4,256)
1940 DIM WORD$(9), WORD2$(3), ID$(8,6), W#(4), RANGE(4), OFFSET(4)
1950 OSC(1)=0 :OSC(2)=0 :SQZ=0 :S=0 :SS=0 :R=0 :PI!=3.14159
1960 RF$="HP.INI"
1970 RETURN
1980 '
1990      *INIT
2000 '
2010 ''  Reading Initialized File  ''
2020 OPEN RF$ FOR INPUT AS #1
2030 LINE INPUT #1, WORD$(0)
2040 LINE INPUT #1, WORD$(1)
2050 FOR I=0 TO 4
2060 INPUT #1, ID$(I,1),ID$(I,2),ID$(I,3),ID$(I,4),ID$(I,5),ID$(I,6)
2070 NEXT I
2080 LINE INPUT #1, WORD$(2)
2090 FOR I=5 TO 8
2100 INPUT #1, WORD$(I-2),ID$(I,1)
2110 NEXT I
2120 INPUT #1, WORD$(7),WORD$(8)
2125 INPUT #1, WORD2$(1),WORD2$(2)
2130 CLOSE #1
2140 PCAD=VAL(ID$(5,1)) :NMB=VAL(ID$(6,1)) :LMT=VAL(ID$(7,1))
2150 RJCT=VAL(ID$(8,1)) :DIR$=WORD$(8)
2155 WTN=VAL(WORD2$(2))
2160 FOR I=1 TO 4
2170 IF LEFT$(ID$(I,1),1)="#" THEN ID$(I,0)="#" ELSE ID$(I,0)="*"
2180 NEXT I
2190 FOR I=2 TO 4
2200 IF ID$(I,0)="*" AND ID$(I-1,0)<>"#" AND ID$(I,2)=ID$(I-1,2) THEN ID$(I,0)="+"
2210 NEXT I
2220 '
2230 ''  Displaying Initialized Data  ''
2240 CLS :CONSOLE ,,,1 :LOCATE 0,2
2250 PRINT WORD$(0) :PRINT WORD$(1)
2260 FOR I=0 TO 4
2270 PRINT ID$(I,0);" ";ID$(I,1);"  ";ID$(I,2);"  ";
2275 PRINT ID$(I,3);"  ";ID$(I,4);"  ";ID$(I,5);"  ";ID$(I,6)
2280 NEXT I
```

```
2290 PRINT WORD$(2)
2300 FOR I=5 TO 8
2310 PRINT WORD$(I-2),ID$(I,1)
2320 NEXT I
2330 PRINT WORD$(7),WORD$(8)
2335 PRINT WORD2$(1),WORD2$(2)
2340 GOSUB *FAQ
2350 IF R=1 THEN R=0 :RETURN *QUIT
2360 '
2370 ''  Getting Initialized Parameter  ''
2380 CLS :LOCATE 0,2
2390 INPUT "PARA-NAME(~3) :",EX$
2400 IF EX$="" THEN EX$="dat"
2402 EX$=LEFT$(EX$,3)
2410 INPUT "START-Z       :",ZO
2420 INPUT "STEP-Z        :",DZ
2425 INPUT "END-Z         :",ZN
2440 '
2450 ''  Enable Keys  ''
2460 KEY ON :STOP ON :HELP ON
2470 ON KEY GOSUB *F1, *F2, *F3, *F4, *F5, *F6, *F7, *F8, *F9, *F10
2480 ON STOP GOSUB *QUIT
2485 ON HELP GOSUB *PAUSE3
2490 '
2500 ''  Set Screen ''
2510 CLS 3 :CONSOLE 2,22,1,1
2520 SCREEN 3,0,0,1
2530 FOR I=1 TO 2
2540 IF ID$(I,0)<>"#" AND ID$(I,3)="1" THEN KEY 1, ID$(I,1)+"CH1"
2550 IF ID$(I,0)<>"#" AND ID$(I,3)="2" THEN KEY 2, ID$(I,1)+"CH2"
2560 IF ID$(I,0)<>"#" AND ID$(I,3)="3" THEN KEY 1, ID$(I,1)+"CH3"
2570 IF ID$(I,0)<>"#" AND ID$(I,3)="4" THEN KEY 2, ID$(I,1)+"CH4"
2580 IF ID$(I,0)="+" THEN KEY 3, ID$(I,1)+"2CH"
2590 IF ID$(I,0)="*" THEN KEY 4, ID$(I,1)+"STP"
2600 IF ID$(I,0)="*" THEN KEY 5, ID$(I,1)+"RUN"
2610 IF ID$(I,0)="*" THEN OSC(1)=VAL(ID$(I,2))
2620 NEXT I
2630 FOR I=3 TO 4
2640 IF ID$(I,0)<>"#" AND ID$(I,3)="1" THEN KEY 6, ID$(I,1)+"CH1"
2650 IF ID$(I,0)<>"#" AND ID$(I,3)="2" THEN KEY 7, ID$(I,1)+"CH2"
2660 IF ID$(I,0)<>"#" AND ID$(I,3)="3" THEN KEY 6, ID$(I,1)+"CH3"
2670 IF ID$(I,0)<>"#" AND ID$(I,3)="4" THEN KEY 7, ID$(I,1)+"CH4"
2680 IF ID$(I,0)="+" THEN KEY 8, ID$(I,1)+"2CH"
2690 IF ID$(I,0)="*" THEN KEY 9, ID$(I,1)+"STP"
2700 IF ID$(I,0)="*" THEN KEY 10, ID$(I,1)+"RUN"
2710 IF ID$(I,0)="*" THEN OSC(2)=VAL(ID$(I,2))
2720 NEXT I
2730 'KEY LIST
```

```
2740 RETURN
2750 *F1 :SQZ=1 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2760 *F2 :SQZ=2 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2770 *F3 :SQZ=3 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2780 *F4 :SQZ=4 :GOSUB *HPSTOP :SQZ=0 :RETURN
2790 *F5 :SQZ=5 :GOSUB *HPRUN :SQZ=0 :RETURN
2800 *F6 :SQZ=6 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2810 *F7 :SQZ=7 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2820 *F8 :SQZ=8 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2830 *F9 :SQZ=9 :GOSUB *HPSTOP :SQZ=0 :RETURN
2840 *F10 :SQZ=10 :GOSUB *HPRUN :SQZ=0 :RETURN
2850 '
2860      *HPSET
2870 '
2880 T$(1)="" :T$(2)="" :TR$(1)="" :TR$(2)=""
2890 FOR I=1 TO 2
2900 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
2910 IF OSC(I)<>0 THEN PRINT@ OSC(I);":SYSTEM:LONGFORM ON;HEADER OFF"
2920 IF OSC(I)<>0 THEN PRINT@ OSC(I);":ACQUIRE:TYPE?"
2925 IF OSC(I)<>0 THEN INPUT@ OSC(I);T$(I) :T$(I)=LEFT$(T$(I),4) :TYPE$=T$(I)
2930 IF OSC(I)<>0 THEN PRINT@ OSC(I);":TIMEBASE:RANGE?"
2935 IF OSC(I)<>0 THEN INPUT@ OSC(I);TR$(I) :TRANGE=VAL(TR$(I))*1E8
2940 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
2950 NEXT I
2960 LOCATE 32,2 :PRINT SPACE$(47) :LOCATE 32,2
2970 IF OSC(1)*OSC(2)<>0 AND T$(1)<>T$(2) THEN PRINT "SET TIME-ADJUSTs OF HP545s AS THE SAME
TYPE !" :COLOR@ (32,2)-(79,2),6
2975 IF OSC(1)*OSC(2)<>0 AND TR$(1)<>TR$(2) THEN PRINT "SET TIME-ADJUSTs OF HP545s AS THE SAME
TYPE !" :COLOR@ (32,2)-(79,2),6
2980 IF OSC(1)*OSC(2)<>0 AND T$(1)<>T$(2) THEN GOSUB *ALERT :RETURN *DATANM
2990 IF TYPE$="AVER" THEN TPOINTS=500 :VPOINTS=1024 :FORMAT$="WORD"
3000 IF TYPE$="NORM" THEN TPOINTS=8000 :VPOINTS=256 :FORMAT$="COMPRESSED"
3010 RETURN
3020 '
3030      *HPSTOP
3040 '
3050 IO=1 :IN=2
3060 IF SQZ=1 THEN IO=1 :IN=1
3070 IF SQZ=9 THEN IO=2 :IN=2
3080 FOR I=IO TO IN
3090 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
3100 IF OSC(I)<>0 THEN PRINT@ OSC(I);":STOP"
3110 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
3120 NEXT I
3130 RETURN
3140 '
3150      *HPRUN
3160 '
```

```
3170 IO=1 :IN=2
3180 IF SQZ=5 THEN IO=1 :IN=1
3190 IF SQZ=10 THEN IO=2 :IN=2
3200 FOR I=IO TO IN
3210 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
3220 IF OSC(I)<>0 THEN PRINT@ OSC(I);":ERASE PMEMORY0"
3230 IF OSC(I)<>0 THEN PRINT@ OSC(I);":RUN"
3240 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
3250 NEXT I
3260 RETURN
3270 '
3280      *HPSEND
3290 '
3300 GOSUB *HPSTOP
3310 IO=1 :IN=4
3320 IF SQZ=1 THEN IO=1 :IN=1
3330 IF SQZ=2 THEN IO=2 :IN=2
3340 IF SQZ=3 THEN IO=1 :IN=2
3350 IF SQZ=6 THEN IO=3 :IN=3
3360 IF SQZ=7 THEN IO=4 :IN=4
3370 IF SQZ=8 THEN IO=3 :IN=4
3380 FOR I=IO TO IN
3390 IF ID$(I,0)<>"#" THEN OS=VAL(ID$(I,2))
3400 IF ID$(I,0)<>"#" THEN WBYTE &H3F,&H20+OS,&H1;
3410 'IF ID$(I,0)<>"#" THEN PRINT@ OS;":STORE CHANNEL"+ID$(I,3)+",WMEMORY"+ID$(I,3)
3420 'IF ID$(I,0)<>"#" THEN PRINT@ OS;":WAVEFORM:SOURCE WMEMORY"+ID$(I,3)
3425 IF ID$(I,0)<>"#" THEN PRINT@ OS;":WAVEFORM:SOURCE CHANNEL"+ID$(I,3)
3430 IF ID$(I,0)<>"#" THEN PRINT@ OS;":CHANNEL"+ID$(I,3)+":RANGE?"
3440 IF ID$(I,0)<>"#" THEN INPUT@ OS;RAN$ :RANGE(I)=VAL(RAN$)*1000
3450 IF ID$(I,0)<>"#" THEN PRINT@ OS;":CHANNEL"+ID$(I,3)+":OFFSET?"
3460 IF ID$(I,0)<>"#" THEN INPUT@ OS;OF$ :OFFSET(I)=VAL(OF$)*1000
3470 IF ID$(I,0)<>"#" THEN PRINT@ OS;":WAVEFORM:FORMAT "+FORMAT$+";DATA?"
3480 IF ID$(I,0)<>"#" THEN RBYTE &H3F,&H40+OS,&H20+PCAD;PD1,PD2
3490 IF ID$(I,0)<>"#" THEN FOR J=1 TO PD2-&H30 :RBYTE ;D :NEXT J
3500 IF ID$(I,0)<>"#" AND TYPE$="NORM" THEN FOR J=0 TO TPOINTS-1 :RBYTE ;WD(I-1,J)
3510 IF ID$(I,0)<>"#" AND TYPE$="NORM" THEN NEXT J
3520      IF      ID$(I,0)<>"#"      AND      TYPE$="AVER"      THEN      FOR      J=0      TO
TPOINTS-1 :RBYTE ;WD1,WD2 :WD(I-1,J)=WD1*8+WD2/32
3530 IF ID$(I,0)<>"#" AND TYPE$="AVER" THEN NEXT J
3540 IF ID$(I,0)<>"#" THEN RBYTE ;DD :WBYTE &H5F;
3550 IF ID$(I,0)<>"#" THEN WBYTE &H3F,&H20+OS,&H1;
3560 NEXT I
3570 GOSUB *HPRUN
3580 RETURN
3590 '
3600      *WARNING
3610 '
3612 MIN(1)=VPOINTS :MIN(2)=VPOINTS :MIN(3)=VPOINTS :MIN(4)=VPOINTS
```

```
3614 WW(1)=0 :WW(2)=0 :WW(3)=0 :WW(4)=0
3615 MAX(1)=0 :MAX(2)=0 :MAX(3)=0 :MAX(4)=0
3616 WN(1)=0 :WN(2)=0 :USCH$="" :OSCH$=""
3618 FOR I=1 TO 4 :FOR J=0 TO TPOINTS-1
3620 IF ID$(I,0)<>"#" AND ID$(I,6)="Y" AND MIN(I)>WD(I-1,J) THEN MIN(I)=WD(I-1,J)
3622 IF ID$(I,0)<>"#" AND ID$(I,6)="Y" AND MAX(I)<WD(I-1,J) THEN MAX(I)=WD(I-1,J)
3623 IF ID$(I,0)="#" OR ID$(I,6)="N" THEN WW(I)=1 :J=TPOINTS-1
3624 NEXT J :NEXT I
3626 FOR I=1 TO 4
3628 IF WW(I)=0 AND MIN(I)>VPOINTS/2-LMT/4 AND MAX(I)<VPOINTS/2+LMT/4 AND RANGE(I)>MVR*4 THEN
WN(1)=1 ELSE USCH$=USCH$+"  "
3630 IF WW(I)=0 AND MIN(I)>VPOINTS/2-LMT/4 AND MAX(I)<VPOINTS/2+LMT/4 AND RANGE(I)>MVR*4 THEN
USCH$=USCH$+STR$(I)
3632 IF WW(I)=0 AND (MIN(I)<LMT OR MAX(I)>VPOINTS-LMT) THEN WN(2)=1 ELSE OSCH$=OSCH$+"  "
3634 IF WW(I)=0 AND (MIN(I)<LMT OR MAX(I)>VPOINTS-LMT) THEN OSCH$=OSCH$+STR$(I)
3638 NEXT I
3640 LOCATE 50,0 :PRINT SPACE$(29)
3642 IF WN(1)=1 THEN LOCATE 50,0 :PRINT "UNDER:";USCH$
3644 IF WN(2)=1 THEN LOCATE 66,0 :PRINT "OVER:";OSCH$
3646 IF WN(1)+WN(2)<>0 THEN COLOR@ (50,0)-(79,0),2
3648 RETURN
3650 '
3660      *RMS
3670 '
3680 Z=L-1000 :KK=K+1
3690 IF KK>23 THEN KK=23
3700 LOCATE 0,KK :PRINT SF2$;"  ";
3710 FOR I=1 TO 4 :AV!=0 :RM!=0
3720  IF  ID$(I,0)<>"#"  THEN  FOR  J=RJCT  TO  TPOINTS-RJCT-1  :AV!=AV!+WD(I-1,J)  :NEXT
J :AD(I)=CINT(AV!/(TPOINTS-2*RJCT))
3730     IF     ID$(I,0)<>"#"     AND     ID$(I,4)="Y"     THEN     FOR     J=RJCT     TO
TPOINTS-RJCT-1 :RM!=RM!+(WD(I-1,J)-AD(I))^2 :NEXT J
3740          IF          ID$(I,0)<>"#"          AND          ID$(I,4)="Y"          THEN
RD0!(I,M)=SQR(RM!/(TPOINTS-2*RJCT))*RANGE(I)/VPOINTS
3750 IF ID$(I,0)<>"#" AND ID$(I,4)="Y" THEN PRINT CSNG(CINT(RD0!(I,M)*10))/10;
3760 IF ID$(I,0)<>"#" AND ID$(I,4)="N" THEN ADD!=(AD(I)-VPOINTS/2)*RANGE(I)/VPOINTS+OFFSET(I)
3770 IF ID$(I,0)<>"#" AND ID$(I,4)="N" THEN PRINT CSNG(CINT(ADD!*10))/10;
3780 NEXT I
3785 IF WN(1)+WN(2)<>0 THEN COLOR@ (0,KK)-(4,KK),2
3787 PRINT
3790 'FOR I=1 TO 4
3795 'IF Z>=0 THEN RD!(I,Z)=0
3800  'IF  ID$(I,0)<>"#"  AND  M=NMB  AND  ID$(I,4)="Y"  AND  Z>=0  THEN  FOR  J=1  TO
NMB :RD!(I,Z)=RD!(I,Z)+RD0!(I,J)
3805 'IF ID$(I,0)<>"#" AND M=NMB AND ID$(I,4)="Y" AND Z>=0 THEN NEXT J :RD!(I,Z)=RD!(I,Z)/NMB
3810 'NEXT I
3820 RETURN
3830 '
```

```
3840       *SPECT
3850 '
3860 FMAX#=.000001
3870 FOR I=1 TO 4
3880 FOR J=0 TO 1023 :W#=1
3890 IF ID$(I,0)<>"#" AND J<102 THEN W#=.5*(COS(PI*J/102-PI)+1)
3900 IF ID$(I,0)<>"#" AND J>921 THEN W#=.5*(COS(PI*(J-921)/102)+1)
3910 IF ID$(I,0)<>"#" AND TYPE$="NORM" THEN FDR#(J)=W#*(WD(I-1,1000+J)-AD(I)) :FDI#(J)=0
3920 IF ID$(I,0)<>"#" AND TYPE$="AVER" AND J<500 THEN FDR#(J)=W#*(WD(I-1,J)-AD(I)) :FDI#(J)=0
3930 IF ID$(I,0)<>"#" AND TYPE$="AVER" AND J>=500 THEN FDR#(J)=0 :FDI#(J)=0
3940 NEXT J
3950 IF ID$(I,0)<>"#" THEN GOSUB *FFT
3960 FOR J=0 TO 256
3970 IF ID$(I,0)<>"#" THEN FD#(I,J)=(FDR#(J)^2+FDI#(J)^2)/1024
3980 IF ID$(I,0)<>"#" AND FD#(I,J)>FMAX# THEN FMAX#=FD#(I,J)
3990 NEXT J: NEXT I
4000 RETURN
4010 '
4020      *WVDPLY
4030 '
4040 IO=1 :IN=4
4050 IF SQZ=1 THEN IO=1 :IN=1
4060 IF SQZ=2 THEN IO=2 :IN=2
4070 IF SQZ=3 THEN IO=1 :IN=2
4080 IF SQZ=6 THEN IO=3 :IN=3
4090 IF SQZ=7 THEN IO=4 :IN=4
4100 IF SQZ=8 THEN IO=3 :IN=4
4110 SCREEN ,0,0,1
4120 LOCATE 32,1 :PRINT SPACE$(48) :LOCATE 32,1
4130 PRINT "mV/div:"
4140 FOR I=IO TO IN
4150 IF ID$(I,0)<>"#" THEN VIEW (254,84*(I-1)+34)-(637,84*(I-1)+115),0,4
4160 IF ID$(I,0)<>"#" THEN WINDOW (0,0)-(TPOINTS-1,VPOINTS-1)
4170    IF    ID$(I,0)<>"#"    AND    TYPE$="NORM"    THEN    FOR    J=1    TO    7    :LINE
(TPOINTS/8*J-1,0)-(TPOINTS/8*J-1,VPOINTS-1),4,,&HF00F :NEXT J
4180    IF    ID$(I,0)<>"#"    AND    TYPE$="AVER"    THEN    FOR    J=1    TO    9    :LINE
(TPOINTS/10*J-1,0)-(TPOINTS/10*J-1,VPOINTS-1),4,,&HF00F :NEXT J
4190       IF       ID$(I,0)<>"#"       THEN       FOR       J=1       TO       3       :LINE
(0,VPOINTS/4*J-1)-(TPOINTS-1,VPOINTS/4*J-1),4,,&HF00F :NEXT J
4200 IF ID$(I,0)<>"#" THEN FOR J=0 TO TPOINTS-1 :PSET(J,VPOINTS-1-WD(I-1,J)) :NEXT J
4210 IF ID$(I,0)<>"#" THEN LINE (RJCT,VPOINTS-1-AD(I))-(TPOINTS-RJCT-1,VPOINTS-1-AD(I)),2
4220 IF ID$(I,0)<>"#" THEN LOCATE I*10+30,1 :PRINT RANGE(I)/4;"(";OFFSET(I);")"
4230 NEXT I
4233 LOCATE 32,2 :PRINT SPACE$(47)
4235 LOCATE 32,2 :PRINT "T-Range [ns/div]:";TRANGE
4240 RETURN
4250 '
4260      *SPDPLY
```

```
4270 '
4280 SCREEN ,0,1,17
4290 FOR I=1 TO 4
4300 IF ID$(I,0)<>"#" AND I<=2 THEN VIEW (196*(I-1)+249,34)-(196*(I-1)+441,196),0,4
4310 IF ID$(I,0)<>"#" AND I>=3 THEN VIEW (196*(I-3)+249,200)-(196*(I-3)+441,362),0,4
4320 IF ID$(I,0)<>"#" THEN WINDOW (0,-FMAX#)-(256,0)
4330       IF       ID$(I,0)<>"#"       THEN       FOR       J=1       TO       24       :LINE
(CINT(256/25*J),-FMAX#)-(CINT(256/25*J),0),4,,&HF0F0 :NEXT J
4340       IF       ID$(I,0)<>"#"       THEN       FOR       J=1       TO       4       :LINE
(CINT(256/5*J),-FMAX#)-(CINT(256/5*J),0),4,, :NEXT J
4350 IF ID$(I,0)<>"#" THEN FOR J=0 TO 255 :LINE (J,-FD#(I,J))-(J+1,-FD#(I,J+1)) :NEXT J
4360 NEXT I
4363 LOCATE 32,2 :PRINT SPACE$(47)
4365 LOCATE 32,2 :PRINT "F-Range [MHz/div]: 50/S"
4370 RETURN
4380 '
4390      *DATANM
4400 '
4403 IF M<M0 THEN M=MN :L=L-DL
4405 IF M>MN THEN M=M0 :L=L+DL
4410 LOCATE 0,0 :PRINT SPACE$(49) :LOCATE 32,0
4420 SF2$=RIGHT$(STR$(L),3)+CHR$(&H60+M)
4430 IF NMB=1 THEN SF2$=RIGHT$(STR$(L),3)
4440 PRINT "DATA: ";SF2$ :COLOR@ (32,0)-(49,0),6
4450 RETURN
4460 '
4470      *WVSAVE
4480 '
4490 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
4500 SFA$=SF2$+"."+EX$ :SF0$=""
4510 PRINT "W-NAME: ";SFA$ :COLOR@ (0,0)-(31,0),5
4520 IF S=0 THEN LOCATE 8,0 :INPUT "",SF0$ :IF SF0$<>"" THEN SFA$=SF0$
4530 SFA$=WORD$(8)+SFA$
4540 OPEN SFA$ FOR OUTPUT AS #1
4550 PRINT #1, "#OST";
4560 FOR I=1 TO 4
4570 IF ID$(I,0)<>"#" AND ID$(I,5)="Y" AND OFFSET(I)<0 THEN PRINT #1, " ";
4575 IF ID$(I,0)<>"#" AND ID$(I,5)="Y" THEN PRINT #1, STR$(OFFSET(I));
4580 NEXT I
4590 PRINT #1,
4600 PRINT #1, "#RNG";
4610 FOR I=1 TO 4
4620 IF ID$(I,0)<>"#" AND ID$(I,5)="Y" THEN PRINT #1, STR$(RANGE(I));
4630 NEXT I
4640 PRINT #1,
4650 FOR J=0 TO TPOINTS-1 :FOR I=1 TO 4
4660 IF ID$(I,0)<>"#" AND ID$(I,5)="Y" THEN PRINT #1, STR$(WD(I-1,J));
4670 NEXT I :PRINT #1, :NEXT J
```

```
4680 CLOSE #1
4690 LOCATE 0,0 :PRINT SPACE$(32)
4700 RETURN
4710 '
4720 '      *RMSSAVE
4730 '
4740 'Z=L-DZ-1000
4750 'IF Z0<Z THEN J0=Z0 :JN=Z
4760 'IF Z0>Z THEN J0=Z :JN=Z0
4770 'LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
4780 'SFB$="rms."+EX$ :SF0$=""
4790 'PRINT "R-NAME: ";SFB$ :COLOR@ (0,0)-(31,0),5 :LOCATE 8,0 :INPUT "",SF0$ :IF SF0$<>"" THEN
SFB$=SF0$
4800 'SFB$=WORD$(8)+SFB$
4810 'OPEN SFB$ FOR OUTPUT AS #1
4820 'FOR J=J0 TO JN STEP CINT(ABS(DZ)) :PRINT #1, J; :FOR I=1 TO 4
4830 'IF ID$(I,0)<>"#" AND ID$(I,4)="Y" THEN PRINT #1, RD!(I,J);
4840 'NEXT I :PRINT #1, :NEXT J
4850 'CLOSE #1
4860 'RETURN
4870 '
4880      *FAQ
4890 '
4900 LOCATE 0,1 :PRINT SPACE$(32) :LOCATE 0,1
4910 IF AT=0 THEN PRINT "SELECT![2/4/6/8/0/./+/*/-/y/n]" :COLOR@ (0,1)-(31,1),4
4920 IF AT=0 THEN A$="" :WHILE A$="" :A$=INKEY$ :WEND
4930 IF AT=1 THEN PRINT "---- FULL AUTOMATION MODE ----" :COLOR@ (0,1)-(31,1),4
4935 IF AT=1 THEN A$="*"
4940 S=0 :R=0 :WN(1)=0 :WN(2)=0              'Input file-name when S=0
4950 IF (A$="2" OR A$="4" OR A$="6" OR A$="8") AND SS=1 THEN M=M-DM :SS=0 :SCREEN ,2,,
4960 IF A$="2" THEN L=L-DL
4970 IF A$="4" THEN M=M-DM
4980 IF A$="6" THEN M=M+DM
4990 IF A$="8" THEN L=L+DL
5000 GOSUB *DATANM
5002 IF A$="0" OR A$="+" OR A$="*" THEN K=K+1
5004 IF L<>LL AND (A$="0" OR A$="+" OR A$="*") THEN GOSUB *TRANS :GOSUB *PAUSE1
5006 IF A$="0" OR A$="+" OR A$="*" THEN GOSUB *HPSET :GOSUB *PAUSE2 :GOSUB *HPSEND
5012 IF A$="0" OR A$="+" OR A$="*" THEN GOSUB *WARNING :GOSUB *RMS
5014 IF A$="0" OR A$="*" THEN GOSUB *WVDPLY
5016 IF A$="+" OR A$="*" THEN S=1
5020 'IF A$="." OR A$="+" OR A$="*" THEN GOSUB *WVSAVE
5025 'IF A$="." OR A$="+" OR A$="*" THEN M=M+DM :SS=1
5030 IF (A$="." OR A$="+" OR A$="*") AND WN(1)+WN(2)<>0 THEN GOSUB *VRANGE
5035 IF (A$="." OR A$="+" OR A$="*") AND WN(1)+WN(2)=0 THEN GOSUB *WVSAVE
5040 IF (A$="." OR A$="+" OR A$="*") AND WN(1)+WN(2)=0 THEN M=M+DM :SS=1
5042 IF (A$="." OR A$="+" OR A$="*") AND WN(1)+WN(2)=0 THEN LL=L
5044 IF A$="-" THEN GOSUB *SPECT :GOSUB *SPDPLY
```

```
5046 IF A$="y" OR A$="Y" OR A$=CHR$(&H0D) THEN R=0 :SCREEN ,2,,
5048 IF A$="n" OR A$="N" THEN R=1
5050 RETURN
5060 '
5070      *ERRORS
5080 '
5090 LOCATE 0,0 :PRINT SPACE$(80) :LOCATE 0,0
5100 PRINT ERR;"ERROR !"
5110 IF ERR=53 THEN PRINT "Read file not found !"
5120 IF ERR=64 THEN PRINT "Disk broken !"
5130 IF ERR=68 THEN PRINT "Save disk full !"
5140 IF ERR=69 THEN PRINT "Read disk bad !"
5150 IF ERR=76 THEN PRINT "Not exist the directory !"
5155 COLOR@ (0,0)-(79,0),2
5160 GOSUB *ALERT :RESUME *QUIT
5170 '
5180      *ALERT
5190 '
5200 LOCATE 0,1 :PRINT SPACE$(32) :LOCATE 0,1
5210 PRINT "PRESS KEY !!" :COLOR@ (0,1)-(31,1),2
5220 BEEP 1 :A$=""
5230 WHILE A$="" :A$=INKEY$ :WEND :BEEP 0
5240 RETURN
5250 '
5260      *FFT
5270 '
5280 O#=PI!/1024 :P=1: Q=1024 :XX(0)=0
5290 FOR II=1 TO 10
5300 O#=O#*2: P=P*2: Q=Q/2
5310 FOR JJ=0 TO P/2-1
5320 X(2*JJ)=XX(JJ)/2
5330 X(2*JJ+1)=(XX(JJ)+1024)/2
5340 XX(2*JJ)=X(2*JJ)
5350 XX(2*JJ+1)=X(2*JJ+1)
5360 FOR KK=0 TO Q-1
5370 A1R#=FDR#(KK+2*JJ*Q)
5380 A1I#=FDI#(KK+2*JJ*Q)
5390 A2R#=FDR#(KK+(2*JJ+1)*Q)
5400 A2I#=FDI#(KK+(2*JJ+1)*Q)
5410 AR#=A1R#-A2R#
5420 AI#=A1I#-A2I#
5430 WR#=COS(O#*KK): WI#=SIN(O#*KK)
5440 FDR#(KK+2*JJ*Q)=A1R#+A2R#
5450 FDI#(KK+2*JJ*Q)=A1I#+A2I#
5460 FDR#(KK+(2*JJ+1)*Q)=AR#*WR#-AI#*WI#
5470 FDI#(KK+(2*JJ+1)*Q)=AI#*WR#+AR#*WI#
5480 NEXT KK: NEXT JJ: NEXT II
5490 FOR II=0 TO 1023
```

```
5500 IF II<X(II) THEN AR#=FDR#(II): AI#=FDI#(II)
5510 IF II<X(II) THEN FDR#(II)=FDR#(X(II)): FDI#(II)=FDI#(X(II))
5520 IF II<X(II) THEN FDR#(X(II))=AR#: FDI#(X(II))=AI#
5530 NEXT II
5540 RETURN
6000 '
6005 '
6010      *PAUSE1
6012 '
6014 WHILE PNT$="" :WEND :PNT$=""
6016 RETURN
6020 '
6025      *PAUSE2
6030 '
6032 IF TYPE$="AVER" THEN LOCATE 0,0 :PRINT SPC(31)
6034 IF TYPE$="AVER" THEN LOCATE 0,0 :PRINT "NOW WAITING :"
6036 IF TYPE$="AVER" THEN COLOR@ (0,0)-(31,0),5
6038 IF TYPE$="AVER" THEN WT=0 :WHILE WTN>=WT :TM$=TIME$
6040 IF TYPE$="AVER" THEN WHILE TM$=TIME$ :WEND
6042 IF TYPE$="AVER" THEN LOCATE 14,0 :PRINT SPC(17)
6044 IF TYPE$="AVER" THEN LOCATE 14,0 :PRINT WT;"/";WTN
6046 IF TYPE$="AVER" THEN WT=WT+1
6048 IF TYPE$="AVER" THEN COLOR@ (0,0)-(31,0),5 :WEND
6050 RETURN
6060 '
6062      *PAUSE3
6064 '
6066 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
6068 PRINT "NOW PAUSING ! PRESS N OR A !" :COLOR@ (0,0)-(31,0),2 :BEEP
6070 C$="" :WHILE C$="" :C$=INKEY$ :WEND
6072 IF C$="n" OR C$="N" THEN AT=0
6074 IF C$="a" OR C$="A" THEN AT=1
6076 LOCATE 0,0 :PRINT SPACE$(32)
6080 RETURN
6100 '
6110      *TRANS                        'Transmitting Data to PC
6115 '
6120 PRINT #2,RIGHT$(STR$(L),3)
6130 RETURN
6200 '
6210      *RECE                         'Receiving Data to PC
6215 '
6230 FOR H=1 TO HN :NEXT H
6240 PNT0$="" :PNT$=""
6250 *LP :PNT0$=INPUT$(1,#2) :PNT$=PNT$+PNT0$ :ZZ=CINT(VAL(PNT$)) 'ZZ Uni :pt
6260 IF AT=0 AND PNT0$=CHR$(&H0D) THEN LOCATE 50,0 :PRINT SPC(29)
6270 IF AT=0 AND PNT0$=CHR$(&H0D) THEN LOCATE 50,0 :PRINT "Rec Z :";ZZ
6275 IF PNT0$=CHR$(&H0D) AND L<>ZZ+1000 THEN M=M0 :L=ZZ+1000
```

```
6280 IF PNTO$=CHR$(&HOD) THEN COM OFF :CLOSE #2
6290 IF PNTO$=CHR$(&HOD) THEN GOSUB *RS232C :RETURN
6300 GOTO *LP
6400 '
6410      *RS232C                      'Ready to Receive Data
6415 '
6420 OPEN "COM:E72XS" AS #2
6430 COM ON :ON COM GOSUB *RECE
6440 RETURN
6500 '
6510      *VRANGE                      'Changing Vertical Ranges
6520 '
6522 IF AT=0 THEN LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
6524 IF AT=0 THEN PRINT "CHANGE VRANGE ?" :COLOR@ (0,0)-(31,0),6 :BEEP
6526 IF AT=0 THEN B$="" :WHILE B$="" :B$=INKEY$ :WEND
6528 IF AT=0 THEN LOCATE 0,0 :PRINT SPACE$(32)
6530 IF AT=0 AND (B$="n" OR B$="N") THEN RETURN
6540 FOR I=1 TO 4
6550 UC$=MID$(USCH$,I*2,1)
6555 OC$=MID$(OSCH$,I*2,1)
6565 IF ID$(I,0)<>"#" THEN OS=VAL(ID$(I,2))
6570 IF ID$(I,0)<>"#" THEN WBYTE &H3F,&H20+OS,&H1;
6575  IF  ID$(I,0)<>"#"  AND  RANGE(I)<MVR*8  AND  RANGE(I)>MVR*4  AND  VAL(UC$)=I  THEN
RANGE(I)=MVR*4*2
6580        IF        ID$(I,0)<>"#"        AND        VAL(UC$)=I        THEN        PRINT@
OS;":CHANNEL"+ID$(I,3)+":RANGE"+STR$(RANGE(I)/1000/2)
6585        IF        ID$(I,0)<>"#"        AND        VAL(OC$)=I        THEN        PRINT@
OS;":CHANNEL"+ID$(I,3)+":RANGE"+STR$(RANGE(I)/1000*2)
6590 IF ID$(I,0)<>"#" THEN WBYTE &H3F,&H20+OS,&H1;
6600 NEXT I
6610 RETURN



1000 '  HP2A.BAS (n88BASIC)   ''''''''''''''''''''''''  Since Jan. 2000  '
1010 '                                                                    '
1020 '  Automatical Measurement Softwear                                  '
1025 '  for HP545 Series with PC98 and GP-IB Board and PDS120-6           '
1030 '                                                                    '
1040 '                                    Programed by TAKEDA Tsuyoshi    '
1050 '                                                                    '
1060 '                                                                    '
1070 '  References: NEC,                                                  '
1080 '              PC-8897 GP-IB Interface Board USER'S MANUAL (1990)    '
1085 '              PC-9801-29N GP-IB Interface Board USER'S MANUAL       '
1090 '                                                                    '
1100 '              K. YAMAGIWA and M. SASAKI,                            '
1110 '              Rep. Fact. Sci. SHIZUOKA UNIV. Vol.26, 31 (1992)      '
```

```
1120 '                                                                      '
1130 '                                                                      '
1140 '  Initial file, "HP.INI".                                             '
1160 '  #Initial File for HP*.BAS                                           '
1170 '  #                                                                   '
1180 '  OSC,ADD,CH,RMS,SAV,WAN                                              '
1190 '  22A, 2,  1,  Y,  Y,  Y :ADD -- Address of OSC (1 - 30)              '
1200 '  22A, 2,  2,  Y,  Y,  Y :CH  -- Channel of OSC (1 - 4)               '
1210 '  10A, 3,  1,  Y,  Y,  Y :RMS -- Root Mean Square (Y or N(average))  '
1220 '  10A, 3,  2,  Y,  Y,  Y :SAV -- Save (Y or N)                        '
1225 '  10A, 3,  2,  Y,  Y,  Y :WAN -- Warn (Y or N)                        '
1230 '  #                                                                   '
1240 '  PC98-ADD, 1     :Address of GPIB-board.                             '
1250 '  RPT-NMB,  2     :Repeated number as the same condition.             '
1260 '  WRN-LMT,  32    :Warned points on scaling from bottom and top.      '
1270 '  H-RJCT,   0     :Rejected points on RMS from left and right.        '
1280 '  SAVE-DIR, B:¥   :Saved directory.                                   '
1285 '  WT-TIME,  7     :Waitting time (sec).                               '
1287 '  PDS-ADD,  5     :Address of PDS120-6.                               '
1290 '                                                                      '
1300 '  Select screen, "SELECT![2/4/6/8/0/./+/*/-/y/n]".                    '
1301 '  2/4  :Back step on data number.                                     '
1302 '  6/8  :Forward step on data number.                                  '
1303 '  0    :Getting and showing data.                                     '
1304 '  .    :Saving data.                                                  '
1305 '  +    :(Next step and) getting and saving data.                      '
1306 '  *    :(Next step and) getting, showing and saving data.             '
1307 '  -    :Showing spectram by FFT.                                      '
1308 '  y    :"Yes" or next.                                                '
1309 '  n    :"No" or exit.                                                 '
1310 '                                                                      '
1312 '  Pressing HELP key, change to Full Automation Mode or Normal Mode.   '
1314 '                                                                      '
1316 '                                                                      '
1320 '  NOTICE!: This program is available for the following conditions:     '
1330 '           Sampling time rate,      1 (ns/points)                     '
1340 '           Max vertical points,     256  or 1024                      '
1350 '           Max horizontal points,   8000 or 500                       '
1360 '                                                                      '
1362 '           Need another controllable PC connected with RS232C.        '
1365 '                                                                      '
1370 ''''''''''''''''''''''''''''''''''''''''''  Last modified in Apr. 2003  '
1380 '
1390 '
1400 ''''''''''''''''''''''''''    MAIN  ROUTINE    ''''''''''''''''''''''''''
1410 '
1420     *MAIN
1430 '
```

```
1440 ON ERROR GOTO *ERRORS
1450 CMD DELIM=0: CMD TIMEOUT=5
1460 ISET IFC :ISET REN
1470 DEFINT A-Z
1480 '
1490 GOSUB *DEFINES :GOSUB *INIT
1510 GOSUB *RS232C :GOSUB *DPSSET
1520 K=0 :L=Z0+1000 :LL=0 :DL=DZ :MO=1 :M=MO :MN=NMB :DM=1
1530 WHILE 1
1580 GOSUB *FAQ
1590 IF R=1 THEN GOSUB *FIN               'Finish Program when R=1
1595 IF AT=1 AND L=ZN+1000 AND V!=VN! AND M>MN THEN GOSUB *QUIT
1635 WEND
1640 '
1650      *FIN
1660 '
1670 'GOSUB *RMSSAVE
1680 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
1690 PRINT "FINISH ?" :COLOR@ (0,0)-(31,0),6 :BEEP
1700 GOSUB *FAQ
1710 IF R=0 THEN GOSUB *QUIT
1730 IF R=1 THEN R=0 :A$="" :GOSUB *DPSSET :RETURN
1750 '
1760      *QUIT
1770 '
1775 R=1 :GOSUB *DPSSET
1780 LOCATE 0,1 :PRINT SPACE$(32) :LOCATE 0,1
1790 PRINT "FINISHED !!" :COLOR@ (0,1)-(31,1),2
1800 BEEP :BEEP :BEEP :BEEP :BEEP
1810 CLOSE :CLEAR :CLS 3 :CONSOLE 0,25,1,1
1820 KEY OFF :STOP OFF
1825 COM OFF :CLOSE #2
1830 ON ERROR GOTO 0
1840 IRESET REN
1850 END
1860 '
1870 ''''''''''''''''''''''''     SUB  ROUTINE    ''''''''''''''''''''''''
1880 '
1890      *DEFINES
1900 '
1902 MVR=4                    'MVR: Min. V-Range (mv/div)
1905 HN=2000                  'HN: Receiving Time to another PC
1906 FOW=100                  'FOW: File open wait
1907 AT=0                     'When AT=1, automation mode runs.
1910 DIM WD(3,8000), AD(4), RD0!(4,10)  ', RD!(4,450)
1920 DIM X(1024), XX(1024), MIN(4) ,MAX(4), WN(2)
1930 DIM FDR#(1024), FDI#(1024), FD#(4,256)
1940 DIM WORD$(9), WORD2$(5), ID$(8,6), W#(4), RANGE(4), OFFSET(4)
```

```
1950 OSC(1)=0 :OSC(2)=0 :SQZ=0 :S=0 :SS=0 :R=0 :PI!=3.14159
1960 RF$="HP.INI" :EX$=".dat"
1970 RETURN
1980 '
1990      *INIT
2000 '
2010 ''  Reading Initialized File  ''
2020 OPEN RF$ FOR INPUT AS #1
2030 LINE INPUT #1, WORD$(0)
2040 LINE INPUT #1, WORD$(1)
2050 FOR I=0 TO 4
2060 INPUT #1, ID$(I,1),ID$(I,2),ID$(I,3),ID$(I,4),ID$(I,5),ID$(I,6)
2070 NEXT I
2080 LINE INPUT #1, WORD$(2)
2090 FOR I=5 TO 8
2100 INPUT #1, WORD$(I-2),ID$(I,1)
2110 NEXT I
2120 INPUT #1, WORD$(7),WORD$(8)
2125 INPUT #1, WORD2$(1),WORD2$(2)
2127 INPUT #1, WORD2$(3),WORD2$(4)
2130 CLOSE #1
2140 PCAD=VAL(ID$(5,1)) :NMB=VAL(ID$(6,1)) :LMT=VAL(ID$(7,1))
2150 RJCT=VAL(ID$(8,1)) :DIR$=WORD$(8)
2155 WTN=VAL(WORD2$(2)) :PDS=VAL(WORD2$(4))
2160 FOR I=1 TO 4
2170 IF LEFT$(ID$(I,1),1)="#" THEN ID$(I,0)="#" ELSE ID$(I,0)="*"
2180 NEXT I
2190 FOR I=2 TO 4
2200 IF ID$(I,0)="*" AND ID$(I-1,0)<>"#" AND ID$(I,2)=ID$(I-1,2) THEN ID$(I,0)="+"
2210 NEXT I
2220 '
2230 ''  Displaying Initialized Data  ''
2240 CLS :CONSOLE ,,,1 :LOCATE 0,2
2250 PRINT WORD$(0) :PRINT WORD$(1)
2260 FOR I=0 TO 4
2270 PRINT ID$(I,0);" ";ID$(I,1);"  ";ID$(I,2);"  ";
2275 PRINT ID$(I,3);"  ";ID$(I,4);"  ";ID$(I,5);"  ";ID$(I,6)
2280 NEXT I
2290 PRINT WORD$(2)
2300 FOR I=5 TO 8
2310 PRINT WORD$(I-2),ID$(I,1)
2320 NEXT I
2330 PRINT WORD$(7),WORD$(8)
2335 PRINT WORD2$(1),WORD2$(2)
2337 PRINT WORD2$(3),WORD2$(4)
2340 GOSUB *FAQ
2350 IF R=1 THEN R=0 :RETURN *QUIT
2360 '
```

```
2370 ''  Getting Initialized Parameter  ''
2380 CLS :LOCATE 0,2
2390 INPUT "PARA-NAME(~3) :",PARA$
2400 IF PARA$<>"" THEN EX$="."+LEFT$(PARA$,3)
2410 INPUT "START-Z       :",Z0
2420 INPUT "STEP-Z        :",DZ
2422 INPUT "END-Z         :",ZN
2424 INPUT "START-V (V)   :",V0!
2426 INPUT "STEP-V  (V)   :",DV!
2428 INPUT "END-V   (V)   :",VN!
2430 V0!=CINT(V0!*10)/10 :DV!=CINT(DV!*10)/10
2435 VN!=CINT(VN!*10)/10 :V!=V0!
2440 '
2450 ''  Enable Keys  ''
2460 KEY ON :STOP ON :HELP ON
2470 ON KEY GOSUB *F1, *F2, *F3, *F4, *F5, *F6, *F7, *F8, *F9, *F10
2480 ON STOP GOSUB *QUIT
2485 ON HELP GOSUB *PAUSE3
2490 '
2500 ''  Set Screen ''
2510 CLS 3 :CONSOLE 2,22,1,1
2520 SCREEN 3,0,0,1
2530 FOR I=1 TO 2
2540 IF ID$(I,0)<>"#" AND ID$(I,3)="1" THEN KEY 1, ID$(I,1)+"CH1"
2550 IF ID$(I,0)<>"#" AND ID$(I,3)="2" THEN KEY 2, ID$(I,1)+"CH2"
2560 IF ID$(I,0)<>"#" AND ID$(I,3)="3" THEN KEY 1, ID$(I,1)+"CH3"
2570 IF ID$(I,0)<>"#" AND ID$(I,3)="4" THEN KEY 2, ID$(I,1)+"CH4"
2580 IF ID$(I,0)="+" THEN KEY 3, ID$(I,1)+"2CH"
2590 IF ID$(I,0)="*" THEN KEY 4, ID$(I,1)+"STP"
2600 IF ID$(I,0)="*" THEN KEY 5, ID$(I,1)+"RUN"
2610 IF ID$(I,0)="*" THEN OSC(1)=VAL(ID$(I,2))
2620 NEXT I
2630 FOR I=3 TO 4
2640 IF ID$(I,0)<>"#" AND ID$(I,3)="1" THEN KEY 6, ID$(I,1)+"CH1"
2650 IF ID$(I,0)<>"#" AND ID$(I,3)="2" THEN KEY 7, ID$(I,1)+"CH2"
2660 IF ID$(I,0)<>"#" AND ID$(I,3)="3" THEN KEY 6, ID$(I,1)+"CH3"
2670 IF ID$(I,0)<>"#" AND ID$(I,3)="4" THEN KEY 7, ID$(I,1)+"CH4"
2680 IF ID$(I,0)="+" THEN KEY 8, ID$(I,1)+"2CH"
2690 IF ID$(I,0)="*" THEN KEY 9, ID$(I,1)+"STP"
2700 IF ID$(I,0)="*" THEN KEY 10, ID$(I,1)+"RUN"
2710 IF ID$(I,0)="*" THEN OSC(2)=VAL(ID$(I,2))
2720 NEXT I
2730 'KEY LIST
2740 RETURN
2750 *F1 :SQZ=1 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2760 *F2 :SQZ=2 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2770 *F3 :SQZ=3 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2780 *F4 :SQZ=4 :GOSUB *HPSTOP :SQZ=0 :RETURN
```

```
2790 *F5 :SQZ=5 :GOSUB *HPRUN :SQZ=0 :RETURN
2800 *F6 :SQZ=6 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2810 *F7 :SQZ=7 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2820 *F8 :SQZ=8 :GOSUB *HPSET :GOSUB *HPSEND :GOSUB *WVDPLY :SQZ=0 :RETURN
2830 *F9 :SQZ=9 :GOSUB *HPSTOP :SQZ=0 :RETURN
2840 *F10 :SQZ=10 :GOSUB *HPRUN :SQZ=0 :RETURN
2850 '
2860      *HPSET
2870 '
2880 T$(1)="" :T$(2)="" :TR$(1)="" :TR$(2)=""
2890 FOR I=1 TO 2
2900 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
2910 IF OSC(I)<>0 THEN PRINT@ OSC(I);":SYSTEM:LONGFORM ON;HEADER OFF"
2920 IF OSC(I)<>0 THEN PRINT@ OSC(I);":ACQUIRE:TYPE?"
2925 IF OSC(I)<>0 THEN INPUT@ OSC(I);T$(I) :T$(I)=LEFT$(T$(I),4) :TYPE$=T$(I)
2930 IF OSC(I)<>0 THEN PRINT@ OSC(I);":TIMEBASE:RANGE?"
2935 IF OSC(I)<>0 THEN INPUT@ OSC(I);TR$(I) :TRANGE=VAL(TR$(I))*1E8
2940 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
2950 NEXT I
2960 LOCATE 32,2 :PRINT SPACE$(47) :LOCATE 32,2
2970 IF OSC(1)*OSC(2)<>0 AND T$(1)<>T$(2) THEN PRINT "SET TIME-ADJUSTs OF HP545s AS THE SAME
TYPE !" :COLOR@ (32,2)-(79,2),6
2975 IF OSC(1)*OSC(2)<>0 AND TR$(1)<>TR$(2) THEN PRINT "SET TIME-ADJUSTs OF HP545s AS THE SAME
TYPE !" :COLOR@ (32,2)-(79,2),6
2980 IF OSC(1)*OSC(2)<>0 AND T$(1)<>T$(2) THEN GOSUB *ALERT :RETURN *DATANM
2990 IF TYPE$="AVER" THEN TPOINTS=500 :VPOINTS=1024 :FORMAT$="WORD"
3000 IF TYPE$="NORM" THEN TPOINTS=8000 :VPOINTS=256 :FORMAT$="COMPRESSED"
3010 RETURN
3020 '
3030      *HPSTOP
3040 '
3050 IO=1 :IN=2
3060 IF SQZ=1 THEN IO=1 :IN=1
3070 IF SQZ=9 THEN IO=2 :IN=2
3080 FOR I=IO TO IN
3090 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
3100 IF OSC(I)<>0 THEN PRINT@ OSC(I);":STOP"
3110 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
3120 NEXT I
3130 RETURN
3140 '
3150      *HPRUN
3160 '
3170 IO=1 :IN=2
3180 IF SQZ=5 THEN IO=1 :IN=1
3190 IF SQZ=10 THEN IO=2 :IN=2
3200 FOR I=IO TO IN
3210 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
```

```
3220 IF OSC(I)<>0 THEN PRINT@ OSC(I);":ERASE PMEMORY0"
3230 IF OSC(I)<>0 THEN PRINT@ OSC(I);":RUN"
3240 IF OSC(I)<>0 THEN WBYTE &H3F,&H20+OSC(I),&H1;
3250 NEXT I
3260 RETURN
3270 '
3280      *HPSEND
3290 '
3300 GOSUB *HPSTOP
3310 IO=1 :IN=4
3320 IF SQZ=1 THEN IO=1 :IN=1
3330 IF SQZ=2 THEN IO=2 :IN=2
3340 IF SQZ=3 THEN IO=1 :IN=2
3350 IF SQZ=6 THEN IO=3 :IN=3
3360 IF SQZ=7 THEN IO=4 :IN=4
3370 IF SQZ=8 THEN IO=3 :IN=4
3380 FOR I=IO TO IN
3390 IF ID$(I,0)<>"#" THEN OS=VAL(ID$(I,2))
3400 IF ID$(I,0)<>"#" THEN WBYTE &H3F,&H20+OS,&H1;
3410 'IF ID$(I,0)<>"#" THEN PRINT@ OS;":STORE CHANNEL"+ID$(I,3)+",WMEMORY"+ID$(I,3)
3420 'IF ID$(I,0)<>"#" THEN PRINT@ OS;":WAVEFORM:SOURCE WMEMORY"+ID$(I,3)
3425 IF ID$(I,0)<>"#" THEN PRINT@ OS;":WAVEFORM:SOURCE CHANNEL"+ID$(I,3)
3430 IF ID$(I,0)<>"#" THEN PRINT@ OS;":CHANNEL"+ID$(I,3)+":RANGE?"
3440 IF ID$(I,0)<>"#" THEN INPUT@ OS;RAN$ :RANGE(I)=VAL(RAN$)*1000
3450 IF ID$(I,0)<>"#" THEN PRINT@ OS;":CHANNEL"+ID$(I,3)+":OFFSET?"
3460 IF ID$(I,0)<>"#" THEN INPUT@ OS;OF$ :OFFSET(I)=VAL(OF$)*1000
3470 IF ID$(I,0)<>"#" THEN PRINT@ OS;":WAVEFORM:FORMAT "+FORMAT$+";DATA?"
3480 IF ID$(I,0)<>"#" THEN RBYTE &H3F,&H40+OS,&H20+PCAD;PD1,PD2
3490 IF ID$(I,0)<>"#" THEN FOR J=1 TO PD2-&H30 :RBYTE ;D :NEXT J
3500 IF ID$(I,0)<>"#" AND TYPE$="NORM" THEN FOR J=0 TO TPOINTS-1 :RBYTE ;WD(I-1,J)
3510 IF ID$(I,0)<>"#" AND TYPE$="NORM" THEN NEXT J
3520      IF      ID$(I,0)<>"#"      AND      TYPE$="AVER"      THEN      FOR      J=0      TO
TPOINTS-1 :RBYTE ;WD1,WD2 :WD(I-1,J)=WD1*8+WD2/32
3530 IF ID$(I,0)<>"#" AND TYPE$="AVER" THEN NEXT J
3540 IF ID$(I,0)<>"#" THEN RBYTE ;DD :WBYTE &H5F;
3550 IF ID$(I,0)<>"#" THEN WBYTE &H3F,&H20+OS,&H1;
3560 NEXT I
3570 GOSUB *HPRUN
3580 RETURN
3590 '
3600      *WARNING
3610 '
3612 MIN(1)=VPOINTS :MIN(2)=VPOINTS :MIN(3)=VPOINTS :MIN(4)=VPOINTS
3614 MAX(1)=0 :MAX(2)=0 :MAX(3)=0 :MAX(4)=0
3615 WW(1)=0 :WW(2)=0 :WW(3)=0 :WW(4)=0
3616 WN(1)=0 :WN(2)=0 :USCH$="" :OSCH$=""
3618 FOR I=1 TO 4 :FOR J=0 TO TPOINTS-1
3620 IF ID$(I,0)<>"#" AND ID$(I,6)="Y" AND MIN(I)>WD(I-1,J) THEN MIN(I)=WD(I-1,J)
```

```
3622 IF ID$(I,0)<>"#" AND ID$(I,6)="Y" AND MAX(I)<WD(I-1,J) THEN MAX(I)=WD(I-1,J)
3623 IF ID$(I,0)="#" OR ID$(I,6)="N" THEN WW(I)=1 :J=TPOINTS-1
3624 NEXT J :NEXT I
3626 FOR I=1 TO 4
3628 IF WW(I)=0 AND MIN(I)>VPOINTS/2-LMT/4 AND MAX(I)<VPOINTS/2+LMT/4 AND RANGE(I)>MVR*4 THEN
WN(1)=1 ELSE USCH$=USCH$+"  "
3630 IF WW(I)=0 AND MIN(I)>VPOINTS/2-LMT/4 AND MAX(I)<VPOINTS/2+LMT/4 AND RANGE(I)>MVR*4 THEN
USCH$=USCH$+STR$(I)
3632 IF WW(I)=0 AND (MIN(I)<LMT OR MAX(I)>VPOINTS-LMT) THEN WN(2)=1 ELSE OSCH$=OSCH$+"  "
3634 IF WW(I)=0 AND (MIN(I)<LMT OR MAX(I)>VPOINTS-LMT) THEN OSCH$=OSCH$+STR$(I)
3638 NEXT I
3640 LOCATE 50,0 :PRINT SPACE$(29)
3642 IF WN(1)=1 THEN LOCATE 50,0 :PRINT "UNDER:";USCH$
3644 IF WN(2)=1 THEN LOCATE 66,0 :PRINT "OVER:";OSCH$
3646 IF WN(1)+WN(2)<>0 THEN COLOR@ (50,0)-(79,0),2
3648 RETURN
3650 '
3660      *RMS
3670 '
3680 Z=L-1000 :KK=K+1
3690 IF KK>23 THEN KK=23
3700 LOCATE 0,KK :PRINT SF2$;SF3$;"  ";
3710 FOR I=1 TO 4 :AV!=0 :RM!=0
3720  IF  ID$(I,0)<>"#"  THEN  FOR  J=RJCT  TO  TPOINTS-RJCT-1  :AV!=AV!+WD(I-1,J)  :NEXT
J :AD(I)=CINT(AV!/(TPOINTS-2*RJCT))
3730     IF     ID$(I,0)<>"#"     AND     ID$(I,4)="Y"     THEN     FOR     J=RJCT     TO
TPOINTS-RJCT-1 :RM!=RM!+(WD(I-1,J)-AD(I))^2 :NEXT J
3740          IF          ID$(I,0)<>"#"          AND          ID$(I,4)="Y"          THEN
RDO!(I,M)=SQR(RM!/(TPOINTS-2*RJCT))*RANGE(I)/VPOINTS
3750 IF ID$(I,0)<>"#" AND ID$(I,4)="Y" THEN PRINT CSNG(CINT(RDO!(I,M)*10))/10;
3760 IF ID$(I,0)<>"#" AND ID$(I,4)="N" THEN ADD!=(AD(I)-VPOINTS/2)*RANGE(I)/VPOINTS+OFFSET(I)
3770 IF ID$(I,0)<>"#" AND ID$(I,4)="N" THEN PRINT CSNG(CINT(ADD!*10))/10;
3780 NEXT I
3785 IF WN(1)+WN(2)<>0 THEN COLOR@ (0,KK)-(8,KK),2
3787 PRINT
3790 'FOR I=1 TO 4
3795 'IF Z>=0 THEN RD!(I,Z)=0
3800  'IF  ID$(I,0)<>"#"  AND  M=NMB  AND  ID$(I,4)="Y"  AND  Z>=0  THEN  FOR  J=1  TO
NMB :RD!(I,Z)=RD!(I,Z)+RDO!(I,J)
3805 'IF ID$(I,0)<>"#" AND M=NMB AND ID$(I,4)="Y" AND Z>=0 THEN NEXT J :RD!(I,Z)=RD!(I,Z)/NMB
3810 'NEXT I
3820 RETURN
3830 '
3840       *SPECT
3850 '
3860 FMAX#=.000001
3870 FOR I=1 TO 4
3880 FOR J=0 TO 1023 :W#=1
```

```
3890 IF ID$(I,0)<>"#" AND J<102 THEN W#=.5*(COS(PI*J/102-PI)+1)
3900 IF ID$(I,0)<>"#" AND J>921 THEN W#=.5*(COS(PI*(J-921)/102)+1)
3910 IF ID$(I,0)<>"#" AND TYPE$="NORM" THEN FDR#(J)=W#*(WD(I-1,1000+J)-AD(I)) :FDI#(J)=0
3920 IF ID$(I,0)<>"#" AND TYPE$="AVER" AND J<500 THEN FDR#(J)=W#*(WD(I-1,J)-AD(I)) :FDI#(J)=0
3930 IF ID$(I,0)<>"#" AND TYPE$="AVER" AND J>=500 THEN FDR#(J)=0 :FDI#(J)=0
3940 NEXT J
3950 IF ID$(I,0)<>"#" THEN GOSUB *FFT
3960 FOR J=0 TO 256
3970 IF ID$(I,0)<>"#" THEN FD#(I,J)=(FDR#(J)^2+FDI#(J)^2)/1024
3980 IF ID$(I,0)<>"#" AND FD#(I,J)>FMAX# THEN FMAX#=FD#(I,J)
3990 NEXT J: NEXT I
4000 RETURN
4010 '
4020      *WVDPLY
4030 '
4040 IO=1 :IN=4
4050 IF SQZ=1 THEN IO=1 :IN=1
4060 IF SQZ=2 THEN IO=2 :IN=2
4070 IF SQZ=3 THEN IO=1 :IN=2
4080 IF SQZ=6 THEN IO=3 :IN=3
4090 IF SQZ=7 THEN IO=4 :IN=4
4100 IF SQZ=8 THEN IO=3 :IN=4
4110 SCREEN ,0,0,1
4120 LOCATE 32,1 :PRINT SPACE$(48) :LOCATE 32,1
4130 PRINT "mV/div:"
4140 FOR I=IO TO IN
4150 IF ID$(I,0)<>"#" THEN VIEW (254,84*(I-1)+34)-(637,84*(I-1)+115),0,4
4160 IF ID$(I,0)<>"#" THEN WINDOW (0,0)-(TPOINTS-1,VPOINTS-1)
4170    IF    ID$(I,0)<>"#"    AND    TYPE$="NORM"    THEN    FOR    J=1    TO    7    :LINE
(TPOINTS/8*J-1,0)-(TPOINTS/8*J-1,VPOINTS-1),4,,&HF00F :NEXT J
4180    IF    ID$(I,0)<>"#"    AND    TYPE$="AVER"    THEN    FOR    J=1    TO    9    :LINE
(TPOINTS/10*J-1,0)-(TPOINTS/10*J-1,VPOINTS-1),4,,&HF00F :NEXT J
4190       IF       ID$(I,0)<>"#"       THEN       FOR       J=1       TO       3       :LINE
(0,VPOINTS/4*J-1)-(TPOINTS-1,VPOINTS/4*J-1),4,,&HF00F :NEXT J
4200 IF ID$(I,0)<>"#" THEN FOR J=0 TO TPOINTS-1 :PSET(J,VPOINTS-1-WD(I-1,J)) :NEXT J
4210 IF ID$(I,0)<>"#" THEN LINE (RJCT,VPOINTS-1-AD(I))-(TPOINTS-RJCT-1,VPOINTS-1-AD(I)),2
4220 IF ID$(I,0)<>"#" THEN LOCATE I*10+30,1 :PRINT RANGE(I)/4;"(";OFFSET(I);")"
4230 NEXT I
4233 LOCATE 32,2 :PRINT SPACE$(47)
4235 LOCATE 32,2 :PRINT "T-Range [ns/div]:";TRANGE
4240 RETURN
4250 '
4260      *SPDPLY
4270 '
4280 SCREEN ,0,1,17
4290 FOR I=1 TO 4
4300 IF ID$(I,0)<>"#" AND I<=2 THEN VIEW (196*(I-1)+249,34)-(196*(I-1)+441,196),0,4
4310 IF ID$(I,0)<>"#" AND I>=3 THEN VIEW (196*(I-3)+249,200)-(196*(I-3)+441,362),0,4
```

```
4320 IF ID$(I,0)<>"#" THEN WINDOW (0,-FMAX#)-(256,0)
4330       IF       ID$(I,0)<>"#"       THEN       FOR       J=1       TO       24       :LINE
(CINT(256/25*J),-FMAX#)-(CINT(256/25*J),0),4,,&HF0F0 :NEXT J
4340       IF       ID$(I,0)<>"#"       THEN       FOR       J=1       TO       4       :LINE
(CINT(256/5*J),-FMAX#)-(CINT(256/5*J),0),4,, :NEXT J
4350 IF ID$(I,0)<>"#" THEN FOR J=0 TO 255 :LINE (J,-FD#(I,J))-(J+1,-FD#(I,J+1)) :NEXT J
4360 NEXT I
4363 LOCATE 32,2 :PRINT SPACE$(47)
4365 LOCATE 32,2 :PRINT "F-Range [MHz/div]: 50/S"
4370 RETURN
4380 '
4390      *DATANM
4400 '
4402 IF M<M0 THEN M=MN :V!=V!-DV!
4404 IF M>MN THEN M=M0 :V!=V!+DV!
4406 IF V!<V0! THEN V!=VN! :L=L-DL
4408 IF V!>VN! THEN V!=V0! :L=L+DL
4410 LOCATE 0,0 :PRINT SPACE$(49) :LOCATE 32,0
4420 SF2$=RIGHT$(STR$(L),3)
4425 SF3$=RIGHT$(STR$(V!*10+10000),4)+CHR$(&H60+M)
4430 IF NMB=1 THEN SF3$=RIGHT$(STR$(V!*10+10000),4)
4440 PRINT "DATA: ";SF2$;SF3$ :COLOR@ (32,0)-(49,0),6
4450 RETURN
4460 '
4470      *WVSAVE
4480 '
4490 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
4500 SFA$=SF2$+SF3$+EX$ :SF0$=""
4510 PRINT "W-NAME: ";SFA$ :COLOR@ (0,0)-(31,0),5
4520 IF S=0 THEN LOCATE 8,0 :INPUT "",SF0$ :IF SF0$<>"" THEN SFA$=SF0$
4530 SFA$=WORD$(8)+SFA$
4540 OPEN SFA$ FOR OUTPUT AS #1
4545 FOR FOW=0 TO FOWN :NEXT FOW
4550 PRINT #1, "#OST";
4560 FOR I=1 TO 4
4570 IF ID$(I,0)<>"#" AND ID$(I,5)="Y" AND OFFSET(I)<0 THEN PRINT #1, " ";
4575 IF ID$(I,0)<>"#" AND ID$(I,5)="Y" THEN PRINT #1, STR$(OFFSET(I));
4580 NEXT I
4590 PRINT #1,
4600 PRINT #1, "#RNG";
4610 FOR I=1 TO 4
4620 IF ID$(I,0)<>"#" AND ID$(I,5)="Y" THEN PRINT #1, STR$(RANGE(I));
4630 NEXT I
4640 PRINT #1,
4650 FOR J=0 TO TPOINTS-1 :FOR I=1 TO 4
4660 IF ID$(I,0)<>"#" AND ID$(I,5)="Y" THEN PRINT #1, STR$(WD(I-1,J));
4670 NEXT I :PRINT #1, :NEXT J
4680 CLOSE #1
```

```
4690 LOCATE 0,0 :PRINT SPACE$(32)
4700 RETURN
4710 '
4720 '      *RMSSAVE
4730 '
4740 'Z=L-DZ-1000
4750 'IF Z0<Z THEN J0=Z0 :JN=Z
4760 'IF Z0>Z THEN J0=Z :JN=Z0
4770 'LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
4780 'SFB$="rms"+EX$ :SF0$=""
4790 'PRINT "R-NAME: ";SFB$ :COLOR@ (0,0)-(31,0),5 :LOCATE 8,0 :INPUT "",SF0$ :IF SF0$<>"" THEN
SFB$=SF0$
4800 'SFB$=WORD$(8)+SFB$
4810 'OPEN SFB$ FOR OUTPUT AS #1
4820 'FOR J=J0 TO JN STEP CINT(ABS(DZ)) :PRINT #1, J; :FOR I=1 TO 4
4830 'IF ID$(I,0)<>"#" AND ID$(I,4)="Y" THEN PRINT #1, RD!(I,J);
4840 'NEXT I :PRINT #1, :NEXT J
4850 'CLOSE #1
4860 'RETURN
4870 '
4880      *FAQ
4890 '
4900 LOCATE 0,1 :PRINT SPACE$(32) :LOCATE 0,1
4910 IF AT=0 THEN PRINT "SELECT![2/4/6/8/0/./+/*/-/y/n]" :COLOR@ (0,1)-(31,1),4
4920 IF AT=0 THEN A$="" :WHILE A$="" :A$=INKEY$ :WEND
4930 IF AT=1 THEN PRINT "---- FULL AUTOMATION MODE ----" :COLOR@ (0,1)-(31,1),4
4935 IF AT=1 THEN A$="*"
4940 S=0 :R=0 :WN(1)=0 :WN(2)=0              'Input file-name when S=0
4950 IF (A$="2" OR A$="4" OR A$="6" OR A$="8") AND SS=1 THEN M=M-DM :SS=0 :SCREEN ,2,,
4960 IF A$="2" THEN L=L-DL
4970 IF A$="4" THEN M=M-DM
4980 IF A$="6" THEN M=M+DM
4990 IF A$="8" THEN L=L+DL
5000 GOSUB *DATANM
5002 IF A$="0" OR A$="+" OR A$="*" THEN K=K+1
5004 IF L<>LL AND (A$="0" OR A$="+" OR A$="*") THEN GOSUB *TRANS :GOSUB *PAUSE1
5006 IF A$="0" OR A$="+" OR A$="*" THEN GOSUB *DPSSET :GOSUB *HPRUN
5008 IF A$="0" OR A$="+" OR A$="*" THEN GOSUB *HPSET :GOSUB *PAUSE2 :GOSUB *HPSEND
5010 IF V!=VN! AND (A$="0" OR A$="+" OR A$="*") THEN VVN!=V! :V!=0 :GOSUB *DPSSET :V!=VVN!
5012 IF A$="0" OR A$="+" OR A$="*" THEN GOSUB *WARNING :GOSUB *RMS
5014 IF A$="0" OR A$="*" THEN GOSUB *WVDPLY
5016 IF A$="+" OR A$="*" THEN S=1
5020 'IF A$="." OR A$="+" OR A$="*" THEN GOSUB *WVSAVE
5025 'IF A$="." OR A$="+" OR A$="*" THEN M=M+DM :SS=1
5030 IF (A$="." OR A$="+" OR A$="*") AND WN(1)+WN(2)<>0 THEN GOSUB *VRANGE
5035 IF (A$="." OR A$="+" OR A$="*") AND WN(1)+WN(2)=0 THEN GOSUB *WVSAVE
5040 IF (A$="." OR A$="+" OR A$="*") AND WN(1)+WN(2)=0 THEN M=M+DM :SS=1
5042 IF (A$="." OR A$="+" OR A$="*") AND WN(1)+WN(2)=0 THEN LL=L
```

```
5044 IF A$="-" THEN GOSUB *SPECT :GOSUB *SPDPLY
5046 IF A$="y" OR A$="Y" OR A$=CHR$(&H0D) THEN R=0 :SCREEN ,2,,
5048 IF A$="n" OR A$="N" THEN R=1
5050 RETURN
5060 '
5070      *ERRORS
5080 '
5090 LOCATE 0,0 :PRINT SPACE$(80) :LOCATE 0,0
5100 PRINT ERR;"ERROR !"
5110 IF ERR=53 THEN PRINT "Read file not found !"
5120 IF ERR=64 THEN PRINT "Disk broken !"
5130 IF ERR=68 THEN PRINT "Save disk full !"
5140 IF ERR=69 THEN PRINT "Read disk bad !"
5150 IF ERR=76 THEN PRINT "Not exist the directory !"
5155 COLOR@ (0,0)-(79,0),2
5160 GOSUB *ALERT :RESUME *QUIT
5170 '
5180      *ALERT
5190 '
5200 LOCATE 0,1 :PRINT SPACE$(32) :LOCATE 0,1
5210 PRINT "PRESS KEY !!" :COLOR@ (0,1)-(31,1),2
5220 BEEP 1 :A$=""
5230 WHILE A$="" :A$=INKEY$ :WEND :BEEP 0
5240 RETURN
5250 '
5260      *FFT
5270 '
5280 O#=PI!/1024 :P=1: Q=1024 :XX(0)=0
5290 FOR II=1 TO 10
5300 O#=O#*2: P=P*2: Q=Q/2
5310 FOR JJ=0 TO P/2-1
5320 X(2*JJ)=XX(JJ)/2
5330 X(2*JJ+1)=(XX(JJ)+1024)/2
5340 XX(2*JJ)=X(2*JJ)
5350 XX(2*JJ+1)=X(2*JJ+1)
5360 FOR KK=0 TO Q-1
5370 A1R#=FDR#(KK+2*JJ*Q)
5380 A1I#=FDI#(KK+2*JJ*Q)
5390 A2R#=FDR#(KK+(2*JJ+1)*Q)
5400 A2I#=FDI#(KK+(2*JJ+1)*Q)
5410 AR#=A1R#-A2R#
5420 AI#=A1I#-A2I#
5430 WR#=COS(O#*KK): WI#=SIN(O#*KK)
5440 FDR#(KK+2*JJ*Q)=A1R#+A2R#
5450 FDI#(KK+2*JJ*Q)=A1I#+A2I#
5460 FDR#(KK+(2*JJ+1)*Q)=AR#*WR#-AI#*WI#
5470 FDI#(KK+(2*JJ+1)*Q)=AI#*WR#+AR#*WI#
5480 NEXT KK: NEXT JJ: NEXT II
```

```
5490 FOR II=0 TO 1023
5500 IF II<X(II) THEN AR#=FDR#(II): AI#=FDI#(II)
5510 IF II<X(II) THEN FDR#(II)=FDR#(X(II)): FDI#(II)=FDI#(X(II))
5520 IF II<X(II) THEN FDR#(X(II))=AR#: FDI#(X(II))=AI#
5530 NEXT II
5540 RETURN
6000 '
6005 '
6010      *PAUSE1
6012 '
6014 WHILE PNT$="" :WEND :PNT$=""
6016 RETURN
6020 '
6025      *PAUSE2
6030 '
6032 IF TYPE$="AVER" THEN LOCATE 0,0 :PRINT SPC(31)
6034 IF TYPE$="AVER" THEN LOCATE 0,0 :PRINT "NOW WAITING :"
6036 IF TYPE$="AVER" THEN COLOR@ (0,0)-(31,0),5
6038 IF TYPE$="AVER" THEN WT=0 :WHILE WTN>=WT :TM$=TIME$
6040 IF TYPE$="AVER" THEN WHILE TM$=TIME$ :WEND
6042 IF TYPE$="AVER" THEN LOCATE 14,0 :PRINT SPC(17)
6044 IF TYPE$="AVER" THEN LOCATE 14,0 :PRINT WT;"/";WTN
6046 IF TYPE$="AVER" THEN WT=WT+1
6048 IF TYPE$="AVER" THEN COLOR@ (0,0)-(31,0),5 :WEND
6050 RETURN
6060 '
6062      *PAUSE3
6064 '
6066 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
6068 PRINT "NOW PAUSING ! PRESS N OR A !" :COLOR@ (0,0)-(31,0),2 :BEEP
6070 C$="" :WHILE C$="" :C$=INKEY$ :WEND
6072 IF C$="n" OR C$="N" THEN AT=0
6074 IF C$="a" OR C$="A" THEN AT=1
6076 LOCATE 0,0 :PRINT SPACE$(32)
6080 RETURN
6100 '
6110      *TRANS                        'Transmitting Data to PC
6115 '
6120 PRINT #2,RIGHT$(STR$(L),3)
6130 RETURN
6200 '
6210      *RECE                         'Receiving Data to PC
6215 '
6230 FOR H=1 TO HN :NEXT H
6240 PNT0$="" :PNT$=""
6250 *LP :PNT0$=INPUT$(1,#2) :PNT$=PNT$+PNT0$ :ZZ=CINT(VAL(PNT$)) 'ZZ Uni :pt
6260 IF AT=0 AND PNT0$=CHR$(&H0D) THEN LOCATE 50,0 :PRINT SPC(29)
6270 IF AT=0 AND PNT0$=CHR$(&H0D) THEN LOCATE 50,0 :PRINT "Rec Z :";ZZ
```

```
6275 IF PNTO$=CHR$(&HOD) AND L<>ZZ+1000 THEN M=M0 :L=ZZ+1000
6280 IF PNTO$=CHR$(&HOD) THEN COM OFF :CLOSE #2
6290 IF PNTO$=CHR$(&HOD) THEN GOSUB *RS232C :RETURN
6300 GOTO *LP
6400 '
6410      *RS232C                        'Ready to Receive Data
6415 '
6420 OPEN "COM:E72XS" AS #2
6430 COM ON :ON COM GOSUB *RECE
6440 RETURN
6500 '
6510      *VRANGE                        'Changing Vertical Ranges
6520 '
6522 IF AT=0 THEN LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
6524 IF AT=0 THEN PRINT "CHANGE VRANGE ?" :COLOR@ (0,0)-(31,0),6 :BEEP
6526 IF AT=0 THEN B$="" :WHILE B$="" :B$=INKEY$ :WEND
6528 IF AT=0 THEN LOCATE 0,0 :PRINT SPACE$(32)
6530 IF AT=0 AND (B$="n" OR B$="N") THEN RETURN
6540 FOR I=1 TO 4
6550 UC$=MID$(USCH$,I*2,1)
6555 OC$=MID$(OSCH$,I*2,1)
6565 IF ID$(I,0)<>"#" THEN OS=VAL(ID$(I,2))
6570 IF ID$(I,0)<>"#" THEN WBYTE &H3F,&H20+OS,&H1;
6575  IF  ID$(I,0)<>"#"  AND  RANGE(I)<MVR*8  AND  RANGE(I)>MVR*4  AND  VAL(UC$)=I  THEN
RANGE(I)=MVR*4*2
6580        IF        ID$(I,0)<>"#"        AND        VAL(UC$)=I        THEN        PRINT@
OS;":CHANNEL"+ID$(I,3)+":RANGE"+STR$(RANGE(I)/1000/2)
6585        IF        ID$(I,0)<>"#"        AND        VAL(OC$)=I        THEN        PRINT@
OS;":CHANNEL"+ID$(I,3)+":RANGE"+STR$(RANGE(I)/1000*2)
6590 IF ID$(I,0)<>"#" THEN WBYTE &H3F,&H20+OS,&H1;
6600 NEXT I
6610 RETURN
6700 '
6710      *DPSSET
6720 '
6730 WBYTE &H3F,&H20+PDS,&H1; :PRINT@ PDS;"SW1"
6750 PRINT@ PDS;"VA"+STR$(V!)
6760 IF R=1 THEN PRINT@ PDS;"VA00.0" :PRINT@ PDS;"SW0"
6770 RETURN
```

```
#Initial File for HP*.EXE
#
OSC,ADD,CH,RMS,SAV,WAN
22A, 2,  1,  N,  Y,  N
22A, 2,  2,  N,  Y,  N
#10A, 3,  1,  N,  Y,  N
```

```
#10A, 3,  2,  N,  Y,  N
#
PC98-ADD, 1
RPT-NMB,  1
WRN-LMT,  256
H-RJCT,   0
SAVE-DIR, b:¥50v290a¥
WT-TIME,  13
PDS-ADD,  5


100 '  MON.BAS (n88BASIC)  ''''''''''''''''''''''''''''''  Since Feb. 2002
102 '
104 ' This is a software to know discharge currents,
106 ' pressures and probe positions z[pt] on experiments.
108 '
110 '                                          Programed by TAKEDA Tsuyoshi
112 '
114 '
116 ' Access speed to I/O on 486/Pentium PC is faster than
118 ' conversion speed on A/D board.
120 ' So, this program needs some weights after starting A/D
121 ' as "OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0
122 ' :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0" on 486 PC.
123 ' (Weights on Pentium PC demand 4 times more than above.)
124 '
126 ' Available for TRANS and RECE ROUTINE by RS232C.
128 ' No using TEST1 and TEST2 ROUTINE together.
130 '
131 '     Meaning the following functions for KEYs:
132 '
134 '         KEY '0':   RESET   ---  Resetting to present z[pt].
136 '         KEY '5':   STOP    ---  Stopping moving.
138 '         KEY '4/6': MOVE    ---  Moving.
140 '         KEY '*':   INPUT   ---  Input z[pt] to move.
142 '
144 '     NOTE:   z[pt] must be absolute values, not relative.
145 '
146 '     Showing the following values:
148 '
150 '         PORT*:    Id[mA]   ---  Discharge current
152 '         PORT*:    P[Torr]  ---  Pressure
154 '         PORT*:    Z[mm,pt] ---  z-position
156 '
160 ''''''''''''''''''''''''''''''''''''''''''''  Last modified in Apr. 2003
210 '
220 '
230 CLS :CLEAR,
```

```
240 STOP ON :ON STOP GOSUB *QUIT
245 LOCATE 10,2 :INPUT "USE RS232C? [Y/N]",RS$
250 IF RS$="y" OR RS$="Y" OR RS$="" THEN R%=1 ELSE R%=0
255 IF R%=1 THEN GOSUB *RS232C
257 'Available for RS232C when R%=1.
300 '
310 '
320 *MAIN
330 '
340 CH1%=2 :CH2%=3 :CH3%=4 :CH4%=5 :CH5%=6     'Channel numbers of A/D
345 CHS1%=2^5 :CHS2%=2^6                       'Channel numbers of A/D
350 IN%=4 :HN%=3000  'IN%: Average num / HN%: Waiting time when RECE is on
355 ZZ%=-1 :PPP4=0 :PPP5=0                     'PPP4/5: Ex-counts on Z0#
360 SS%=0 :SSS%=0 :M1%=0 :M2%=0 :ZPP%=0 :ZPPS%=0 :Z0#=0 :KS%=0
370 PA3%=0 :PA4%=0 :PA5%=0 :PB3%=0 :PB4%=0 :PB5%=0 :PPA4=0 :PPA5=0
380 J%=0 :JN%=CINT(800/IN%)        'JN%: Waiting time on step-by-step mode
390 OUT &HD0,0
400 '
410 LOCATE 10,2 :PRINT SPACE$(50)
420 LOCATE 10,2 :INPUT "STARTING Z[pt] :",ZP            'INP Z[pt]
425 Z0#=ZP*375/2 :ZPP%=CINT(ZP)
430 LOCATE 10,2 :PRINT "KEY: 0/5/4,6/* -- RESET/STOP/MOVE/Z[pt] :"
440 LOCATE 10,4 :PRINT "PORT";CH1%;": Id[mA] ="
450 LOCATE 10,6 :PRINT "PORT";CH2%;": P[Torr]="
460 LOCATE 10,8 :PRINT "PORT";CH3%;": Counter="
470 LOCATE 10,10 :PRINT "        Z[pt,mm]="
500 '
510 '
520 WHILE 1
540 DH1=0 :DL1=0 :DH2=0 :DL2=0 :DH3=0 :DL3=0 :DH4=0 :DL4=0 :DH5=0 :DL5=0
600 '
610 FOR I%=1 TO IN%
620 OUT &HD0,CH1%+SS% :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
622 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
625 DH1=INP(&HD4)+DH1 :DL1=INP(&HD6)+DL1
630 OUT &HD0,CH2%+SS% :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
632 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
635 DH2=INP(&HD4)+DH2 :DL2=INP(&HD6)+DL2
640 OUT &HD0,CH3%+SS% :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
642 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
645 DH3=INP(&HD4)+DH3 :DL3=INP(&HD6)+DL3
650 OUT &HD0,CH4%+SS% :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
652 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
655 DH4=INP(&HD4)+DH4 :DL4=INP(&HD6)+DL4
660 OUT &HD0,CH5%+SS% :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
662 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
665 DH5=INP(&HD4)+DH5 :DL5=INP(&HD6)+DL5
670 GOSUB *MOVE1
```

```
690 NEXT I%
700 '
710 P1=(256*DH1+DL1)/IN% :P2=(256*DH2+DL2)/IN%
720 P3=(256*DH3+DL3)/IN% :P4=(256*DH4+DL4)/IN%
730 P5=(256*DH5+DL5)/IN%
740 GOSUB *COUNT
745 GOSUB *MOVE1
750 X=(P1-2048)*0.0025*10              'AMP=100
760 Y=(P2-2048)*0.0025*2               'GI_REC=1/2
770 Z=Z0#*4/225 :ZP=Z0#*2/375          'Z uni: mm / ZP uni: pt
780 ZZZ%=CINT(ZP-0.5*ZPPS%)
800 IF ZZ%<>-1 THEN GOSUB *MOVE2
810 'IF ZZ%=-1 AND ZPPS%<>0 THEN GOSUB *TEST1     'ONE STEP
820 'IF ZZ%=-1 AND ZPPS%<>0 THEN GOSUB *TEST2     'STEP BY STEP
825 GOSUB *MOVE1
830 LOCATE 27,4 :PRINT SPACE$(10) :LOCATE 27,4 :PRINT CINT(X*10)/10
840 LOCATE 27,6 :PRINT SPACE$(10) :LOCATE 27,6 :PRINT CINT(Y*100)/100
850 LOCATE 27,8 :PRINT SPACE$(20) :LOCATE 27,8 :PRINT Z0# ;J%
860 LOCATE 27,10 :PRINT SPACE$(20) :LOCATE 27,10 :PRINT CINT(ZP*10)/10;
865 PRINT TAB(34) :PRINT CINT(Z*10)/10
870 IF Z<-0.5 AND ZPPS%=-1 THEN SS%=0 :SSS%=0 :ZPPS%=0 :ZZ%=-1 :M2%=0
875 IF Z>450.5 AND ZPPS%=1 THEN SS%=0 :SSS%=0 :ZPPS%=0 :ZZ%=-1 :M2%=0
880 WEND
900 '
910 '
920 *QUIT
930 OUT &HD0,0
940 COM OFF :CLOSE #1
950 STOP OFF :END
960 '
970 *TRANS                             'TRANSMIT DATA TO PC
980 PRINT #1,STR$(ZZZ%)
990 RETURN
1000 '
1010 *RECE                             'RECEIVE DATA FROM PC
1015 FOR H%=1 TO HN% :NEXT H%
1020 PNT0$="" :PNT$=""
1030 *LP :PNT0$=INPUT$(1,#1) :PNT$=PNT$+PNT0$ :ZZ%=CINT(VAL(PNT$)) 'ZZ% Uni :pt
1040 IF PNT0$=CHR$(&H0D) THEN SS%=0 :SSS%=0 :ZPPS%=0
1050 IF PNT0$=CHR$(&H0D) THEN OUT &HD0,SS%
1060 IF PNT0$=CHR$(&H0D) THEN LOCATE 51,2 :PRINT SPC(5)
1070 IF PNT0$=CHR$(&H0D) THEN LOCATE 51,2 :PRINT ZZ% :M2%=0
1080 IF PNT0$=CHR$(&H0D) THEN COM OFF :CLOSE #1
1085 IF PNT0$=CHR$(&H0D) THEN GOSUB *RS232C :RETURN
1090 GOTO *LP
1100 '
1105 *MOVE1
1110 K$="" :K$=INKEY$ :M1%=0
```

```
1112 IF K$="0" THEN LOCATE 28,10 :PRINT SPACE$(30)
1114 IF K$="0" THEN LOCATE 28,10 :INPUT "",ZP                'INP Z[pt]
1116 IF K$="0" THEN Z0#=ZP*375/2 :ZPP%=CINT(ZP)
1118 IF K$="0" THEN SS%=0 :SSS%=0 :ZPPS%=0 :ZZ%=-1 :M1%=1    'RESET
1120 IF K$="5" THEN SS%=0 :SSS%=0
1125 IF K$="5" THEN ZPP%=ZZZ% :ZPPS%=0 :ZZ%=-1 :M2%=0        'KEY5 & S* OFF
1130 IF K$="6" THEN SS%=CHS1% :ZPPS%=1 :ZZ%=-1               'KEY6 & S* ON
1140 IF K$="4" THEN SS%=CHS2% :ZPPS%=-1 :ZZ%=-1              'KEY4 & S* ON
1150 IF K$="*" THEN SS%=0 :SSS%=0 :ZPPS%=0 :M2%=0 :KS%=1
1160 OUT &HD0,SS%
1170 IF K$="*" THEN LOCATE 52,2 :PRINT SPACE$(5)
1180 IF K$="*" THEN LOCATE 52,2 :INPUT "",ZZ%                'INPT Z[pt]
1190 RETURN
1200 '
1210 *MOVE2
1220 IF ZP<ZZ% AND M2%=0 THEN SS%=CHS1% :ZPPS%=1 :M2%=1          'S* ON
1225 IF ZP>ZZ% AND M2%=0 THEN SS%=CHS2% :ZPPS%=-1 :M2%=1         'S* ON
1230 IF ZP=ZZ% AND M2%=0 THEN SS%=0 :ZPPS%=0 :M2%=1
1235 ZZZ%=CINT(ZP-0.5*ZPPS%)
1240 IF R%=1 AND ZZZ%=ZZ% AND M2%=1 AND KS%=0 THEN GOSUB *TRANS  'TRANSMISSION
1245 IF R%=1 AND ZZZ%=ZZ% AND M2%=1 AND KS%=1 THEN KS%=0
1250 IF ZZZ%=ZZ% AND M2%=1 THEN SS%=0 :SSS%=0                    'S* OFF
1260 IF ZZZ%=ZZ% AND M2%=1 THEN ZPP%=ZZZ% :ZPPS%=0 :ZZ%=-1 :M2%=0
1270 OUT &HD0,SS%
1280 RETURN
1300 '
1310 *COUNT
1320 IF P3>3072 THEN PA3%=1 ELSE PA3%=0
1330 IF P4>3072 THEN PA4%=1 :PP4%=1 ELSE PA4%=0 :PP4%=0
1340 IF P5>3072 THEN PA5%=1 :PP5%=1 ELSE PA5%=0 :PP5%=0
1350 IF PA3%-PB3%=-1 THEN PP3%=1 ELSE PP3%=0
1360 IF PA4%-PB4%=-1 THEN PPA4=PPP4 ELSE PPA4=0
1370 IF PA5%-PB5%=-1 THEN PPA5=PPP5 ELSE PPA5=0
1380 Z0#=Z0#+PP3%*(PP4%-PP5%)+PPA4-PPA5
1390 PB3%=PA3% :PB4%=PA4% :PB5%=PA5%
1400 RETURN
1500 '
1510 *TEST1           'ONE STEP MOTION
1513 ZZ%=ZZZ%+ZPPS%
1515 RETURN
1517 '
1520 *TEST2           'STEP BY STEP MOTION
1530 IF ZZZ%=ZPP% THEN ZPP%=ZPP%+ZPPS% :SSS%=SS% :SS%=0 :J%=0
1540 IF SS%=0 THEN J%=J%+1
1550 IF J%=JN% THEN J%=0 :SS%=SSS%
1560 OUT &HD0,SS%
1570 RETURN
1600 '
```

```
1610 *RS232C
1620 OPEN "COM:E72XS" AS #1
1630 COM ON :ON COM GOSUB *RECE                    'RECEPTION
1640 RETURN



1000 '  LAN.BAS (n88BASIC)    ''''''''''''''''''''''''''  Since Apr. 2002  '
1005 '                                                                      '
1010 '  Langmur Probe Measurements for PC98 and A/D Board                   '
1015 '                                                                      '
1020 '                                      Programed by TAKEDA Tsuyoshi    '
1025 '                                                                      '
1030 '                                                                      '
1035 '  References: NEC,                                                    '
1040 '              N88-NihongoBASIC(86)(Ver6.0),                           '
1045 '              Reference Manual / Users Manual.                        '
1050 '                                                                      '
1055 '                                                                      '
1060 '  Initial file, "LAN.INI".                                            '
1062 '  #Initial File for LAN.BAS                                           '
1064 '  #                                                                   '
1066 '  DVR,       4.85:Divided voltage ratio (%).                          '
1068 '  DPR,       2.5 :Disk probe's radius (mm).                           '
1070 '  OUT-R,     50  :Output resistance (Ohm).                            '
1072 '  AMP-G,     300 :Amplifier's gain (time).                            '
1074 '  WT-TIME,   10  :Waitting time (sec).                                '
1076 '  RPT-NMB,   1   :Repeated number as the same condition.              '
1078 '  V-STEP,    0.5 :Steped voltage (V) [more than 0.1].                 '
1080 '  V1,        0   :Volt. to estimate ion curr.                         '
1082 '  V2,        4   :Volt. to estimate elec. temp.                       '
1084 '  V3,        7   :Volt. to estimate saturated curr.                   '
1086 '  VW,        4   :Volt. width at V1, V2, and V3.                      '
1088 '  SAVE-DIR, B:¥ :Directory to save.                                   '
1090 '                                                                      '
1092 '  Select screen (0), "SELECT[0/2/4/6/8/*/i/t/e/m/s/y/n]".             '
1094 '  4/6  :Back or forward step on data number.                          '
1096 '  2/8  :Back or forward step on analying phase.                       '
1098 '  0    :Getting, analyzing and showing data.                          '
1100 '  *    :Getting, analyzing and showing automatically.                 '
1102 '  i    :Setting the grad. of ion curr. on "Select screen (3)".        '
1104 '  t    :Setting the grad. of elec. temp. on "Select screen (3)".      '
1106 '  e    :Setting the grad. of satu. curr. on "Select screen (3)".      '
1108 '  m    :Changing probe position.                                      '
1110 '  s    :Setting measurement condisions.                               '
1112 '  y    :"Yes" or next.                                                '
1114 '  n    :"No" or exit.                                                 '
1116 '                                                                      '
```

```
1118 '  Select screen (1), "SELECT(m)[0/4/5/6/*]".                          '
1120 '  0    :Resetting present probe position (pt).                        '
1122 '  4/6  :Moving probe (s-port on).                                     '
1124 '  5    :Stopping moving probe (s-port off).                           '
1126 '  *    :Setting position to move (pt).                                '
1128 '  ENTR :Return to "Select screen (0)".                                '
1130 '                                                                      '
1132 '  Select screen (2), "SELECT(s)[0/d/o/a/w/r/v]".                      '
1134 '  0    :Showing ch-port signals to adjust offset.                     '
1136 '  d    :Setting DPR.                                                  '
1138 '  o    :Setting OUT-R.                                                '
1140 '  a    :Setting AMP-G.                                                '
1142 '  w    :Setting WT-TIME.                                              '
1144 '  r    :Setting RPT-NMB.                                              '
1146 '  v    :Setting V-STEP.                                               '
1147 '  ENTR :Return to "Select screen (0)".                                '
1148 '                                                                      '
1150 '  Select screen (3), "SELECT(i/t/e)[0/2/4/6/8/+]".                    '
1152 '  0    :Moving line to white or yellow plots.                         '
1154 '  4/6  :Moving line to left or right.                                 '
1156 '  2/8  :Making width between marked points narrow or broad.           '
1157 '  +    :Showing plasma conditions.                                    '
1158 '  ENTR :Return to "Select screen (0)".                                '
1160 '                                                                      '
1162 '  Pressing HELP key,                                                  '
1164 '  changeable to automation-mode or normal-mode.                       '
1166 '                                                                      '
1170 '                                                                      '
1180 '  NOTICE!  This program is available for the following conditions:     '
1190 '           Min. voltage of ch-port,          -5.1175                  '
1195 '           Max. voltage of ch-port,           5.12                    '
1200 '           Max. horizontal initial points,   32750                   '
1205 '           Max. horizontal plotable points,  1000                    '
1210 '                                                                      '
1220 ''''''''''''''''''''''''''''''''''''''''''  Last modified in Apr. 2003  '
1230 '
1240 '
1400 ''''''''''''''''''''''''    MAIN  ROUTINE    ''''''''''''''''''''''''
1410 '
1420      *MAIN
1430 '
1435 CLEAR,
1440 ON ERROR GOTO *ERRORS
1470 DEFINT A-Z
1480 '
1490 GOSUB *DEFINES :GOSUB *INIT :GOSUB *MON0
1520 L=LZ0+1000 :LL=0 :DL=DLZ :M0=1 :M=M0 :MN=5 :DM=1
1525 GOSUB *DATANM
```

```
1530 WHILE 1
1540 GOSUB *MON
1580 GOSUB *FAQ
1590 IF R=1 THEN GOSUB *FIN               'Finish Program when R=1
1595 IF AT=1 AND L=LZN+1000 AND M>MN THEN GOSUB *QUIT
1635 WEND
1640 '
1650      *FIN
1660 '
1680 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
1690 PRINT "FINISH ?" :COLOR@ (0,0)-(31,0),6 :BEEP
1700 GOSUB *FAQ
1710 IF R=0 THEN GOSUB *QUIT
1730 IF R=1 THEN R=0 :RETURN
1750 '
1760      *QUIT
1770 '
1775 OUT &HD0,0
1780 LOCATE 0,1 :PRINT SPACE$(32) :LOCATE 0,1
1790 PRINT "FINISHED !!" :COLOR@ (0,1)-(31,1),2
1800 BEEP :BEEP :BEEP :BEEP :BEEP
1810 CLOSE :CLEAR :CLS 3 :CONSOLE 0,24,1,1
1820 STOP OFF
1830 ON ERROR GOTO 0
1850 END
1860 '
1870 ''''''''''''''''''''''''''    SUB  ROUTINE    ''''''''''''''''''''''''''
1880 '
1890      *DEFINES
1900 '
1905 AT=0                              'When AT=1, automation mode runs.
1930 DIM DO0(16000), DO1(16000), DO!(2002), DI!(2002), D!(2002)
1935 DIM DDI!(2002), DD!(2002), LD1!(2002), LD2!(2002)
1940 DIM WORD$(13), ID$(13)
1945 EE!=1.187E13
1950 SQZ=0 :S=0 :SC=0 :R=0 :PLSM=1   'When PLSM=1, plasma exists.
1960 RF$="lan.ini" :EX$=".dat"
1970 RETURN
1980 '
1990      *INIT
2000 '
2010 ''  Reading Initialized File  ''
2020 OPEN RF$ FOR INPUT AS #1
2030 LINE INPUT #1, WORD$(0)
2040 LINE INPUT #1, WORD$(1)
2050 FOR I=2 TO 13
2100 INPUT #1, WORD$(I),ID$(I)
2110 NEXT I
```

```
2130 CLOSE #1
2140 DVR!=VAL(ID$(2)) :DPR!=VAL(ID$(3)) :OPR!=VAL(ID$(4)) :AG!=VAL(ID$(5))
2150 WT=VAL(ID$(6)) :RN=VAL(ID$(7)) :DV!=VAL(ID$(8))
2160 V1!=VAL(ID$(9)) :V2!=VAL(ID$(10)) :V3!=VAL(ID$(11)) :DIR$=ID$(13)
2170 VW1!=VAL(ID$(12)) :VW2!=VAL(ID$(12)) :VW3!=VAL(ID$(12))
2180 TVS=1 :EVS=-1
2220 '
2230 ''  Displaying Initialized Data  ''
2240 CLS :CONSOLE ,,,1 :LOCATE 0,3
2250 PRINT WORD$(0) :PRINT WORD$(1)
2300 FOR I=2 TO 13
2310 PRINT WORD$(I),ID$(I)
2320 NEXT I
2340 GOSUB *FAQ
2350 IF R=1 THEN R=0 :RETURN *QUIT
2360 '
2370 ''  Getting Initialized Parameter  ''
2380 CLS :LOCATE 0,3
2390 INPUT "PARA-NAME(~4) :",PARA$
2400 IF PARA$="" THEN PARA$="1000"
2402 PARA$=LEFT$(PARA$,4)
2410 INPUT "START-Z       :",LZ0
2420 INPUT "STEP-Z        :",DLZ
2425 INPUT "END-Z         :",LZN
2437 SF1$=PARA$
2440 '
2450 ''  Enable Keys  ''
2460 KEY ON :STOP ON :HELP ON
2480 ON STOP GOSUB *QUIT
2490 ON HELP GOSUB *PAUSE2
2500 '
2510 ''  Set Screen ''
2520 CLS 3 :SCREEN 3,0,0,1
2525 LOCATE 0,10 :PRINT "[SET]"
2530 FOR I=3 TO 8
2540 LOCATE 0,I+8 :PRINT WORD$(I);TAB(10);ID$(I)
2550 NEXT I
2560 RETURN
3000 '
3010      *MEAS
3020 '
3030 CH1=0 :CH2=1 :CHS1=2^4                    'Channel numbers of A/D ports
3040 ''
3050 FOR H=1 TO RN
3060 OUT &HD0,0 :I=0 :J=0 :IN=2                'IN: Average numbers of CH2
3065 LOCATE 0,0 :PRINT "Now Detecting !" :COLOR@ (0,0)-(15,0),5
3067 '
3070 D1A!=0 :D1B!=0 :D1C!=0 :D2A!=0 :D2B!=0
```

```
3080 OUT &HD0,CH1+CHS1 :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3085 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3090 DH1!=INP(&HD4) :DL1!=INP(&HD6)
3100 D1A!=256*DH1!+DL1!
3110 DH2!=0 :DL2!=0
3120 FOR I=1 TO IN
3130 OUT &HD0,CH2+CHS1 :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3135 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3140 DH2!=INP(&HD4)+DH2! :DL2!=INP(&HD6)+DL2!
3150 NEXT I
3160 D2A!=256*DH2!+DL2!
3170 OUT &HD0,CH1+CHS1 :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3175 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3180 DH1!=INP(&HD4) :DL1!=INP(&HD6)
3190 D1B!=256*DH1!+DL1!
3200 TM$=TIME$ :T=0
3210 WHILE T<=WT
3220 DH2!=0 :DL2!=0
3230 FOR I=1 TO IN
3240 OUT &HD0,CH2+CHS1 :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3245 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3250 DH2!=INP(&HD4)+DH2! :DL2!=INP(&HD6)+DL2!
3260 NEXT I
3270 D2B!=256*DH2!+DL2!
3280 OUT &HD0,CH1+CHS1 :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3285 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
3290 DH1!=INP(&HD4) :DL1!=INP(&HD6)
3300 D1C!=256*DH1!+DL1!
3310 DO0(J)=CINT((D1A!+D1B!+D1C!)/3) :DO1(J)=CINT((D2A!+D2B!)/IN/2)
3320 D1A!=D1B! :D1B!=D1C! :D2A!=D2B! :J=J+1
3325 TIM$=TIME$
3330 IF TM$<>TIM$ THEN LOCATE 15,0 :PRINT SPACE$(15)
3332 IF TM$<>TIM$ THEN LOCATE 15,0 :PRINT T;"/";WT;"(";H;")"
3334 IF TM$<>TIM$ THEN COLOR@ (0,0)-(30,0),5 :TM$=TIME$ :T=T+1
3340 WEND
3345 '
3350 OUT &HD0,0 :IN=J-1
3355 LOCATE 0,0 :PRINT SPACE$(30)
3360 VMIN!=1E30 :VMAX!=-1E30 :IMIN!=1E30 :IMAX!=-1E30
3370 FOR J=0 TO IN
3380 D1!=(DO0(J)-2048)*2.5/48.5
3385 D2!=(DO1(J)-2048)*2500/OPR!/AG!
3390 IF VMIN!>D1! THEN VMINP=J :VMIN!=D1!
3395 IF VMAX!<D1! THEN VMAXP=J :VMAX!=D1!
3400 IF IMIN!>D2! THEN IMINP=J :IMIN!=D2!
3405 IF IMAX!<D2! THEN IMAXP=J :IMAX!=D2!
3410 NEXT J
3415 LOCATE 0,18 :PRINT "[PORTS]"
```

```
3420 LOCATE 0,19 :PRINT SPACE$(20)
3425 LOCATE 0,19 :PRINT "Total Pt.";IN
3430 LOCATE 0,20 :PRINT SPACE$(20)
3435 LOCATE 0,20 :PRINT "ch0 Min. ";(DO0(VMINP)-2048)*0.0025
3440 LOCATE 0,21 :PRINT SPACE$(20)
3445 LOCATE 0,21 :PRINT "ch0 Max. ";(DO0(VMAXP)-2048)*0.0025
3450 LOCATE 0,22 :PRINT SPACE$(20)
3455 LOCATE 0,22 :PRINT "ch1 Min. ";(DO1(IMINP)-2048)*0.0025
3460 LOCATE 0,23 :PRINT SPACE$(20)
3465 LOCATE 0,23 :PRINT "ch1 Max. ";(DO1(IMAXP)-2048)*0.0025
3470 GVSTEP!=5
3475 D1!=(DO0(0)-2048)*2.5/48.5 :D2!=(DO0(IN)-2048)*2.5/48.5
3480 FOR I=-120 TO 40 STEP 5
3485 IF I-5<D1! AND I=>D1! THEN GVMIN0!=I
3490 IF I<=D2! AND I+5>D2! THEN GVMAX0!=I
3495 NEXT I
3500 IF H=1 THEN GVMIN!=GVMIN0! :GVMAX!=GVMAX0!
3510 IF GVMAX!>GVMAX0! THEN GVMAX!=GVMAX0!
3515 IF GVMIN!<GVMIN0! THEN DV0=CINT((GVMIN0!-GVMIN!)/DV!)
3520 IF GVMIN!<GVMIN0! THEN GVMIN!=GVMIN0! :GOSUB *COMP0
3530 GVPOINT=INT((GVMAX!-GVMIN!)/GVSTEP!)
3535 I=0 :J=0 :K=0
3540 DV00!=GVMIN!
3545 D1!=0 :D2!=0 :D2A!=0 :D2B!=0
3550 WHILE DV00!<GVMAX!-DV!
3560 DV1!=GVMIN!+DV!*J :DV2!=GVMIN!+DV!*(J+1)
3570 D1!=(DO0(I)-2048)*2.5/48.5 :D2!=(DO1(I)-2048)*2500/OPR!/AG!
3580 IF DV1!<=D1! AND DV2!>D1! THEN D2B!=D2B!+D2! :K=K+1
3590 IF K=0 AND DV2!<=D1! THEN XX!=DV1!+DV!/2 :GOSUB *COMP1
3600 IF K=0 AND DV2!<=D1! THEN D2B!=YY!*2500/OPR!/AG! :K=1
3610 IF DV2!<=D1! THEN D2B!=D2B!/K
3620 IF J<>0 AND DV2!<=D1! THEN DO!(J)=DV1! :DV00!=DV1!
3625 IF H=1 AND J<>0 AND DV2!<=D1! THEN DI!(J)=(D2A!+D2B!)/2
3627 IF H>1 AND J<>0 AND DV2!<=D1! THEN DI!(J)=DI!(J)+(D2A!+D2B!)/2
3630 IF DV2!<=D1! THEN D2A!=D2B! :D2B!=0
3635 IF DV2!<=D1! THEN I=I-1: J=J+1 :K=0
3640 I=I+1
3642 WEND :
3644 VPOINT=J :DO!(J)=DV2!
3646 IF H=1 THEN DI!(J)=D2A! ELSE DI!(J)=DI!(J)+D2A!
3650 NEXT H
3655 ''
3660 IMIN!=1E30 :IMAX!=-1E30
3670 FOR I=1 TO VPOINT
3675 IF I<3 OR I>VPOINT-2 THEN D!(I)=DI!(I)/RN
3680 IF I=>3 AND I<=VPOINT-2 THEN GOSUB *COMP2 :D!(I)=YY!/RN
3700 IF IMIN!>D!(I) THEN IMIN!=D!(I)
3705 IF IMAX!<D!(I) THEN IMAX!=D!(I)
```

```
3710 NEXT I
3715 LIM1!=LOG(ABS(IMIN!)+1E-10)/LOG(10)
3720 LIM2!=LOG(ABS(IMAX!)+1E-10)/LOG(10)
3725 IF LIM1!>LIM2! THEN LIMIN!=LIM2! :LIMAX!=LIM1!
3730 IF LIM1!<=LIM2! THEN LIMIN!=LIN1! :LIMAX!=LIM2!
3740 FOR J=-6 TO 6
3750 IF J<LIMAX! AND J+0.5=>LIMAX! THEN GISTEP!=(10^J)/5 :LI=J
3760 IF J+0.5<LIMAX! AND J+1=>LIMAX! THEN GISTEP!=(10^J)/2 :LI=J
3770 NEXT J
3780 IF IMIN!*IMAX!>0 AND LIMAX!-LIMIN!<1 THEN GISTEP!=(10^LI)/10
3785 IF IMIN!*IMAX!>0 AND LIMAX!-LIMIN!<0.5 THEN GISTEP!=(10^LI)/20
3790 IF IMIN!*IMAX!>0 AND LIMAX!-LIMIN!<0.1 THEN GISTEP!=(10^LI)/50
3795 D1!=IMIN! :D2!=IMAX! :J=0
3800 FOR II!=-1*10^(LI+1) TO 10^(LI+1) STEP GISTEP!
3810 IF II!-GISTEP!<=D1! AND II!>D1! THEN GIMIN!=II!-GISTEP! :J=1
3820 IF II!<D2! AND II!+GISTEP!=>D2! THEN GIMAX!=II!+GISTEP! :GIPOINT=J+1
3830 J=J+1
3840 NEXT II!
3860 RETURN
3870 '
3872      *DIFF
3874 '
3876 FOR I=2 TO VPOINT-1
3878 DDI!(I)=(D!(I+1)-D!(I-1))/(DO!(I+1)-DO!(I-1))
3880 NEXT I
3882 DDI!(1)=DDI!(2) :DDI!(VPOINT)=DDI!(VPOINT-1)
3884 DIMIN!=1E30 :DIMAX!=-1E30
3886 FOR I=1 TO VPOINT
3888 IF I<3 OR I>VPOINT-2 THEN DD!(I)=DDI!(I)
3890 IF I=>3 AND I<=VPOINT-2 THEN GOSUB *COMP3 :DD!(I)=YY!
3892 IF I=>3 AND I<=VPOINT-2 AND DIMIN!>DD!(I) THEN DIMIN!=DD!(I) :SPP=I
3894 IF I=>3 AND I<=VPOINT-2 AND DIMAX!<DD!(I) THEN DIMAX!=DD!(I)
3896 NEXT I
3898 IF ABS(DIMAX!)<=ABS(DIMIN!) THEN DDA!=IMIN!/DIMIN!
3900 IF ABS(DIMAX!)>ABS(DIMIN!) THEN DDA!=IMAX!/DIMAX!
3902 RETURN
3904 '
3906      *COMPO
3908 '
3910 FOR I=1 TO VPOINT-DVO
3912 DO!(I)=DO!(I+DVO) :DI!(I)=DI!(I+DVO)
3914 NEXT I
3916 RETURN
3918 '
3920      *COMP1
3922 '
3924 YY!=(XX!-(DOO(I)-2048)*2.5/48.5)/(DOO(I-1)-DOO(I))/2.5*48.5
3926 YY!=YY!*(DO1(I-1)-DO1(I))+DO1(I)-2048
```

```
3928 RETURN
3930 '
3932      *COMP2                   'Gram Function
3934 '
3936 YY!=-3*DI!(I-2)+12*DI!(I-1)+17*DI!(I)+12*DI!(I+1)-3*DI!(I+2)
3938 YY!=YY!/35
3940 RETURN
3942 '
3944      *COMP3                   'Gram Function
3946 '
3948 YY!=-3*DDI!(I-2)+12*DDI!(I-1)+17*DDI!(I)+12*DDI!(I+1)-3*DDI!(I+2)
3950 YY!=YY!/35
3952 RETURN
3954 '
3956      *COMP4
3958 '
3960 YY!=AA!*XX!+BB!
3962 RETURN
3964 '
3966      *DDPLY
3968 '
3970 SCREEN ,0,0,1 :GOSUB *GNCLS
3974 VIEW (157,37)-(637,397),0,4
3976 WINDOW (GVMIN!,GIMIN!)-(GVMAX!,GIMAX!)
3978 LINE (GVMIN!,0)-(GVMAX!,0),4 :LINE (0,GIMIN!)-(0,GIMAX!),4
3980 GGI!=(GIMAX!-GIMIN!)/360*2 :GGV!=(GVMAX!-GVMIN!)/480*2
3981 FOR I=1 TO GVPOINT-1
3982 GVE!=(GVMIN!+GVSTEP!*I) MOD 10
3983 IF GVE!=0 THEN GGIO!=GGI! ELSE GGIO!=GGI!/2
3984 LINE (GVMIN!+GVSTEP!*I,GIMIN!)-(GVMIN!+GVSTEP!*I,GIMIN!+GGIO!),4
3986 LINE (GVMIN!+GVSTEP!*I,-GGIO!)-(GVMIN!+GVSTEP!*I,GGIO!),4
3988 LINE (GVMIN!+GVSTEP!*I,GIMAX!-GGIO!)-(GVMIN!+GVSTEP!*I,GIMAX!),4
3990 NEXT I
3992 FOR I=1 TO GIPOINT-1
3994 LINE (GVMIN!,GIMIN!+GISTEP!*I)-(GVMIN!+GGV!,GIMIN!+GISTEP!*I),4
3996 LINE (-GGV!,GIMIN!+GISTEP!*I)-(GGV!,GIMIN!+GISTEP!*I),4
3998 LINE (GVMAX!-GGV!,GIMIN!+GISTEP!*I)-(GVMAX!,GIMIN!+GISTEP!*I),4
4000 NEXT I
4001 FOR I=1 TO VPOINT
4002 PSET(DO!(I),D!(I)) :PSET(DO!(I),DD!(I)*DDA!),5
4003 NEXT I
4004 LINE (DO!(SPP),GIMIN!)-(DO!(SPP),GIMAX!),5
4005 GVSTEPN=1
4006 FOR I=0 TO GVPOINT-1 STEP GVSTEPN
4010 GVE!=(GVMIN!+GVSTEP!*I) MOD 20
4012 IF GVE!=0 THEN LOCATE 18+CINT(60/GVPOINT*I),1 :PRINT CINT(GVMIN!+GVSTEP!*I);
4014 NEXT I
4016 GISTEPN=1
```

```
4018 FOR I=1 TO GIPOINT-1 STEP GISTEPN
4020 GIN!=CINT((GIMIN!+GISTEP!*I)/(10^LI)*1000)/1000
4022 GIE!=(GIMIN!+GISTEP!*I)*100 MOD GISTEP!*100*2
4026 IF GIE!=0 THEN LOCATE 20,CINT(1.5+22.5/GIPOINT*I) :PRINT GIN!;
4028 NEXT I
4030 LOCATE 55,0 :PRINT "Vp (V), Ip ( x";10^LI;"uA)"
4032 RETURN
4034 '
4036      *IDPLY0
4038 '
4040 FOR I=1 TO VPOINT
4042 IF DO!(I)<=V1!-VW1!/2 THEN IVP1=I
4044 IF DO!(I)<=V1!+VW1!/2 THEN IVP2=I
4046 NEXT I
4048 IF IVP2-IVP1<2 THEN IVP2=IVP1+2
4050 IVP0=0 :IVPW=0
4052 '
4054      *IDPLY
4056 '
4058 IVP1=IVP1+IVP0-IVPW :IVP2=IVP2+IVP0+IVPW
4060 IF IVP2-IVP1<2 THEN IVP1=IVP01 :IVP2=IVP02
4062 IF IVP1<1 THEN IVP1=1
4064 IF IVP2>VPOINT THEN IVP2=VPOINT
4066 XX1!=DO!(IVP1) :YY1!=D!(IVP1)
4068 XX2!=DO!(IVP2) :YY2!=D!(IVP2)
4070 V1!=(XX2!+XX1!)/2 :VW1!=XX2!-XX1!
4072 AA1!=(YY2!-YY1!)/(XX2!-XX1!) :AA!=AA1!
4074 BB1!=(XX2!*YY1!-XX1!*YY2!)/(XX2!-XX1!) :BB!=BB1!
4076 XX!=GVMIN! :GOSUB *COMP4 :GI1!=YY!
4078 XX!=GVMAX! :GOSUB *COMP4 :GI2!=YY!
4080 LINE (GVMIN!,GI1!)-(GVMAX!,GI2!),6
4082 CIRCLE (XX1!,YY1!),GGV!,6
4084 CIRCLE (XX2!,YY2!),GGV!,6
4086 IVP01=IVP1 :IVP02=IVP2
4088 RETURN
4134 '
4136      *TDPLY0
4138 '
4140 FOR I=1 TO VPOINT
4142 IF DO!(I)<=V2!-VW2!/2 THEN TVP1=I
4144 IF DO!(I)<=V2!+VW2!/2 THEN TVP2=I
4146 NEXT I
4148 IF TVP2-TVP1<2 THEN TVP2=TVP1+2
4150 TVP0=0 :TVPW=0
4152 '
4154      *TDPLY
4156 '
4158 TVP1=TVP1+TVP0-TVPW :TVP2=TVP2+TVP0+TVPW
```

```
4160 IF TVP2-TVP1<2 THEN TVP1=TVPO1 :TVP2=TVPO2
4162 IF TVP1<1 THEN TVP1=1
4164 IF TVP2>VPOINT THEN TVP2=VPOINT
4166 XX1!=DO!(TVP1) :XX2!=DO!(TVP2)
4167 IF TVS=-1 THEN YY1!=LD1!(TVP1) :YY2!=LD1!(TVP2)
4168 IF TVS=1 THEN YY1!=LD2!(TVP1) :YY2!=LD2!(TVP2)
4170 V2!=(XX2!+XX1!)/2 :VW2!=XX2!-XX1!
4172 AA2!=(YY2!-YY1!)/(XX2!-XX1!) :AA!=AA2!
4174 BB2!=(XX2!*YY1!-XX1!*YY2!)/(XX2!-XX1!) :BB!=BB2!
4176 XX!=GVMIN! :GOSUB *COMP4 :GI1!=YY!
4178 XX!=GVMAX! :GOSUB *COMP4 :GI2!=YY!
4180 LINE (GVMIN!,GI1!)-(GVMAX!,GI2!),2
4182 CIRCLE (XX1!,YY1!),GGV!,2
4184 CIRCLE (XX2!,YY2!),GGV!,2
4186 TVPO1=TVP1 :TVPO2=TVP2
4188 RETURN
4234 '
4236      *EDPLYO
4238 '
4240 FOR I=1 TO VPOINT
4242 IF DO!(I)<=V3!-VW3!/2 THEN EVP1=I
4244 IF DO!(I)<=V3!+VW3!/2 THEN EVP2=I
4246 NEXT I
4248 IF EVP2-EVP1<2 THEN EVP2=EVP1+2
4250 EVPO=0 :EVPW=0
4252 '
4254      *EDPLY
4256 '
4258 EVP1=EVP1+EVPO-EVPW :EVP2=EVP2+EVPO+EVPW
4260 IF EVP2-EVP1<2 THEN EVP1=EVPO1 :EVP2=EVPO2
4262 IF EVP1<1 THEN EVP1=1
4264 IF EVP2>VPOINT THEN EVP2=VPOINT
4266 XX1!=DO!(EVP1) :XX2!=DO!(EVP2)
4267 IF EVS=-1 THEN YY1!=LD1!(EVP1) :YY2!=LD1!(EVP2)
4268 IF EVS=1 THEN YY1!=LD2!(EVP1) :YY2!=LD2!(EVP2)
4270 V3!=(XX2!+XX1!)/2 :VW3!=XX2!-XX1!
4272 AA3!=(YY2!-YY1!)/(XX2!-XX1!) :AA!=AA3!
4274 BB3!=(XX2!*YY1!-XX1!*YY2!)/(XX2!-XX1!) :BB!=BB3!
4276 XX!=GVMIN! :GOSUB *COMP4 :GI1!=YY!
4278 XX!=GVMAX! :GOSUB *COMP4 :GI2!=YY!
4280 LINE (GVMIN!,GI1!)-(GVMAX!,GI2!),1
4282 CIRCLE (XX1!,YY1!),GGV!,1
4284 CIRCLE (XX2!,YY2!),GGV!,1
4286 EVPO1=EVP1 :EVPO2=EVP2
4288 RETURN
4300 '
4302      *PLASMA
4304 '
```

```
4305 IF ABS(AA2!-AA3!)<1E-35 THEN PLSM=0 ELSE PLSM=1
4306 IF PLSM<>0 THEN VVP!=(BB2!-BB3!)/(AA3!-AA2!)
4308 IF PLSM<>0 THEN IIP!=(AA3!*BB2!-AA2!*BB3!)/(AA3!-AA2!) :SG=SGN(IIP!)
4310 IF PLSM<>0 THEN IIP!=10^(ABS(IIP!)+LIMID)
4312 IF PLSM<>0 THEN TE!=1/ABS(AA2!) :NE!=EE!*IIP!/DPR!^2/TE!^0.5
4314 LOCATE 0,18 :PRINT "[PLASMA]"
4316 LOCATE 0,19 :PRINT SPACE$(20) :LOCATE 0,19
4318 IF PLSM<>0 THEN PRINT "Ip (uA) ";IIP!*SG ELSE PRINT "*** N G ***"
4320 LOCATE 0,20 :PRINT SPACE$(20) :LOCATE 0,20
4322 IF PLSM<>0 THEN PRINT "Vp (eV) ";VVP! ELSE PRINT "*** N G ***"
4324 LOCATE 0,21 :PRINT SPACE$(20) :LOCATE 0,21
4326 IF PLSM<>0 THEN PRINT "Te (eV) ";TE! ELSE PRINT "*** N G ***"
4328 LOCATE 0,22 :PRINT SPACE$(20) :LOCATE 0,22
4330 IF PLSM<>0 THEN PRINT "Ne (/m3)";NE! ELSE PRINT "*** N G ***"
4332 LOCATE 0,23 :PRINT SPACE$(20)
4334 RETURN
4500 '
4502      *SLOG
4504 '
4506 LIMIN!=1E30
4508 FOR I=1 TO VPOINT
4510 IF D!(I)<>0 THEN LD0!=LOG(ABS(D!(I)))/LOG(10)
4512 IF D!(I)<>0 AND LIMIN!>LD0! THEN LIMIN!=LD0!
4520 NEXT I
4522 LIMID=INT(LIMIN!) :LIMIN!=1E30 :LIMAX!=-1E30
4524 FOR I=1 TO VPOINT
4526 IF D!(I)>0 THEN LD1!(I)=LOG(D!(I))/LOG(10)-LIMID
4528 IF D!(I)=0 THEN LD1!(I)=0
4530 IF D!(I)<0 THEN LD1!(I)=-(LOG(ABS(D!(I)))/LOG(10)-LIMID)
4532 IF LIMIN!>LD1!(I) THEN LIMIN!=LD1!(I)
4534 IF LIMAX!<LD1!(I) THEN LIMAX!=LD1!(I)
4536 AA!=AA1! :BB!=BB1! :XX!=D0!(I) :GOSUB *COMP4 :LD0!=D!(I)-YY!
4538 IF LD0!>10^LIMID THEN LD2!(I)=LOG(LD0!)/LOG(10)-LIMID
4540 IF LD0!=>-10^LIMID AND LD0!<=10^LIMID THEN LD2!(I)=0
4542 IF LD0!<-10^LIMID THEN LD2!(I)=-(LOG(ABS(LD0!))/LOG(10)-LIMID)
4544 IF LIMIN!>LD2!(I) THEN LIMIN!=LD2!(I)
4546 IF LIMAX!<LD2!(I) THEN LIMAX!=LD2!(I)
4548 NEXT I
4550 GLIMIN!=INT(LIMIN!) :GLIMAX!=INT(LIMAX!)+1
4552 GLIPOINT=INT(GLIMAX!-GLIMIN!) :GLISTEP!=1
4554 RETURN
4560 '
4562      *LDDPLY
4564 '
4566 SCREEN ,0,0,1 :GOSUB *GNCLS
4570 VIEW (157,37)-(637,397),0,4
4572 WINDOW (GVMIN!,GLIMIN!)-(GVMAX!,GLIMAX!)
4574 LINE (GVMIN!,0)-(GVMAX!,0),4 :LINE (0,GLIMIN!)-(0,GLIMAX!),4
```

```
4576 GGLI!=(GLIMAX!-GLIMIN!)/360*2 :GGV!=(GVMAX!-GVMIN!)/480*2
4577 FOR I=1 TO GVPOINT-1
4578 GVE!=(GVMIN!+GVSTEP!*I) MOD 10
4579 IF GVE!=0 THEN GGLI0!=GGLI! ELSE GGLI0!=GGLI!/2
4580 LINE (GVMIN!+GVSTEP!*I,GLIMIN!)-(GVMIN!+GVSTEP!*I,GLIMIN!+GGLI0!),4
4582 LINE (GVMIN!+GVSTEP!*I,-GGLI0!)-(GVMIN!+GVSTEP!*I,GGLI0!),4
4584 LINE (GVMIN!+GVSTEP!*I,GLIMAX!-GGLI0!)-(GVMIN!+GVSTEP!*I,GLIMAX!),4
4586 NEXT I
4588 FOR I=1 TO GLIPOINT
4590 IF I<GLIPOINT THEN LINE (GVMIN!,GLIMIN!+GLISTEP!*I)-(GVMIN!+GGV!,GLIMIN!+GLISTEP!*I),4
4592 IF I<GLIPOINT THEN LINE (-GGV!,GLIMIN!+GLISTEP!*I)-(GGV!,GLIMIN!+GLISTEP!*I),4
4594 IF I<GLIPOINT THEN LINE (GVMAX!-GGV!,GLIMIN!+GLISTEP!*I)-(GVMAX!,GLIMIN!+GLISTEP!*I),4
4596 GLIN=CINT(GLIMIN!+I)
4598 FOR J=2 TO 9
4600 IF GLIN>0 THEN GLL!=GLIMIN!+GLISTEP!*(I-1)+LOG(J)/LOG(10)
4602 IF GLIN<=0 THEN GLL!=GLIMIN!+GLISTEP!*I-LOG(J)/LOG(10)
4604 LINE (GVMIN!,GLL!)-(GVMIN!+GGV!/2,GLL!),4
4606 LINE (-GGV!/2,GLL!)-(GGV!/2,GLL!),4
4608 LINE (GVMAX!-GGV!/2,GLL!)-(GVMAX!,GLL!),4
4610 NEXT J :NEXT I
4612 FOR I=1 TO VPOINT
4614 PSET(DO!(I),LD1!(I)) :PSET(DO!(I),LD2!(I)),6
4616 NEXT I
4618 LINE (DO!(SPP),GLIMIN!)-(DO!(SPP),GLIMAX!),5
4620 GVSTEPN=1
4622 FOR I=0 TO GVPOINT-1 STEP GVSTEPN
4624 GVE!=(GVMIN!+GVSTEP!*I) MOD 20
4626 IF GVE!=0 THEN LOCATE 18+CINT(60/GVPOINT*I),1 :PRINT CINT(GVMIN!+GVSTEP!*I);
4628 NEXT I
4630 FOR I=1 TO GLIPOINT-1
4632 GLIN=CINT(GLIMIN!+I)
4634 IF GLIN<=0 THEN GIN!=-10^(LIMID-GLIN)
4636 IF GLIN>0 THEN GIN!=10^(LIMID+GLIN)
4638 LOCATE 20,CINT(1.5+22.5/GLIPOINT*I) :PRINT GIN!;
4640 NEXT I
4642 LOCATE 60,0 :PRINT "Vp (V), Ip (uA)"
4644 RETURN
4700 '
4710      *GNCLS
4712 '
4714 LOCATE 55,0 :PRINT SPACE$(24)
4716 LOCATE 18,1 :PRINT SPACE$(60)
4718 FOR I=2 TO 23 :LOCATE 20,I :PRINT SPACE$(8); :NEXT I
4720 LOCATE 54,0 :PRINT SPACE$(25)
4722 RETURN
4800 '
4802      *DATANM
4804 '
```

```
4806 IF M<M0 THEN M=MN :L=L-DL
4808 IF M>MN THEN M=M0 :L=L+DL
4810 LOCATE 0,0 :PRINT SPACE$(49) :LOCATE 32,0
4812 SF2$=RIGHT$(STR$(L),3)
4814 PRINT "DATA: ";SF1$;SF2$;"  #";M :COLOR@ (32,0)-(49,0),6
4816 RETURN
4818 '
4820      *DSAVE
4822 '
4824 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
4826 SFA$=SF1$+SF2$+EX$ :SF0$=""
4828 PRINT "D-NAME: ";SFA$ :COLOR@ (0,0)-(31,0),5
4830 IF S=0 THEN LOCATE 8,0 :INPUT "",SF0$
4832 IF S=0 AND SF0$<>"" THEN SFA$=SF0$
4834 IF S=0 AND (SF0$="n" OR SF0$="N") THEN LOCATE 0,0 :PRINT SPACE$(32)
4836 IF S=0 AND (SF0$="n" OR SF0$="N") THEN RETURN
4838 SFA$=DIR$+SFA$
4840 OPEN SFA$ FOR OUTPUT AS #1
4842 PRINT #1, "# Vp(eV)=";VVP!
4844 PRINT #1, "# Te(eV)=";TE!
4846 PRINT #1, "# Ne(/m3)=";NE!
4848 FOR I=1 TO VPOINT
4850 PRINT #1, STR$(D0!(I));TAB(8);STR$(D!(I));
4852 PRINT #1, TAB(20);STR$(DD!(I)) :NEXT I
4854 CLOSE #1
4856 LOCATE 0,0 :PRINT SPACE$(32)
4858 RETURN
4870 '
4875      *FAQ
4880 '
4885 LOCATE 0,1 :PRINT SPACE$(17) :LOCATE 0,1
4890 IF AT=0 THEN PRINT "SELECT[0/2/4/6/8]" :COLOR@ (0,1)-(17,1),4
4900 LOCATE 0,2 :PRINT SPACE$(20) :LOCATE 0,2
4910 IF AT=0 THEN PRINT "[*/i/t/e/m/s/y/n]" :COLOR@ (0,2)-(17,2),4
4920 IF AT=0 AND A$<>"*" THEN A$="" :WHILE A$="" :A$=INKEY$ :WEND
4925 IF AT=1 THEN LOCATE 0,1 :PRINT SPACE$(17) :LOCATE 0,1
4930 IF AT=1 THEN PRINT "--- FULL AUTO ---" :COLOR@ (0,1)-(17,1),2
4935 IF AT=1 THEN LOCATE 0,2 :PRINT SPACE$(20)
4940 IF AT=1 THEN A$="*"
4945 S=0 :R=0                                        'Input file-name when S=0
4950 IF (A$="2" OR A$="8") THEN M=M0                 'CLEAR SCREEN when SC=1
4955 IF (A$="2" OR A$="8") AND SC=1 THEN SC=0 :SCREEN ,2,, :GOSUB *GNCLS
4960 IF A$="2" THEN L=L-DL
4970 IF A$="4" THEN M=M-DM
4980 IF A$="6" THEN M=M+DM
4990 IF A$="8" THEN L=L+DL
5000 GOSUB *DATANM
5010 IF (A$="0" OR A$="*") AND L<>LL AND M=M0 THEN ZZ=L-1000 :GOSUB *PAUSE1
```

```
5012 IF (A$="0" OR A$="*") AND M=M0 THEN GOSUB *MEAS :GOSUB *DIFF
5014 IF (A$="0" OR A$="*") AND M=M0 THEN GOSUB *DDPLY
5016 IF (A$="0" OR A$="*") AND M=M0+DM*1 THEN GOSUB *DDPLY :GOSUB *IDPLY0
5018 IF (A$="0" OR A$="*") AND M=M0+DM*2 THEN GOSUB *SLOG :GOSUB *LDDPLY
5020 IF (A$="0" OR A$="*") AND M=M0+DM*3 THEN GOSUB *LDDPLY
5022 IF (A$="0" OR A$="*") AND M=M0+DM*3 THEN GOSUB *TDPLY0 :GOSUB *EDPLY0
5024 IF A$="*" THEN S=1
5026 IF (A$="0" OR A$="*") AND M=MN THEN GOSUB *PLASMA
5030 IF (A$="0" OR A$="*") AND M=MN AND PLSM<>0 THEN GOSUB *DSAVE
5032 IF (A$="0" OR A$="*") THEN LL=L :SC=1
5034 IF (A$="i" OR A$="I") AND M=M0+DM*1 THEN GOSUB *SETCON3
5036 IF (A$="t" OR A$="T") AND M=M0+DM*3 THEN GOSUB *SETCON4
5038 IF (A$="e" OR A$="E") AND M=M0+DM*3 THEN GOSUB *SETCON5
5042 IF A$="m" OR A$="M" THEN GOSUB *SETCON1
5044 IF A$="s" OR A$="S" THEN GOSUB *SETCON2
5046 IF A$="y" OR A$="Y" OR A$=CHR$(&H0D) THEN R=0 :SCREEN ,2,,
5048 IF A$="y" OR A$="Y" OR A$=CHR$(&H0D) THEN GOSUB *GNCLS
5050 IF A$="n" OR A$="N" THEN R=1
5052 IF AT=0 AND M=MN THEN A$=""
5054 IF AT=1 AND M=MN AND PLSM=0 THEN L=L-DL :PLSM=1
5056 IF A$="*" THEN M=M+DM
5058 RETURN
5060 '
5070      *ERRORS
5080 '
5090 LOCATE 0,0 :PRINT SPACE$(80) :LOCATE 0,0
5100 PRINT ERR;"ERROR !"
5110 IF ERR=53 THEN PRINT "Read file not found !"
5120 IF ERR=64 THEN PRINT "Disk broken !"
5130 IF ERR=68 THEN PRINT "Save disk full !"
5140 IF ERR=69 THEN PRINT "Read disk bad !"
5150 IF ERR=76 THEN PRINT "Not exist the directory !"
5155 COLOR@ (0,0)-(79,0),2
5160 GOSUB *ALERT :RESUME *QUIT
5170 '
5180      *ALERT
5190 '
5200 LOCATE 0,1 :PRINT SPACE$(32) :LOCATE 0,1
5210 PRINT "PRESS KEY !!" :COLOR@ (0,1)-(31,1),2
5220 BEEP 1 :A$=""
5230 WHILE A$="" :A$=INKEY$ :WEND :BEEP 0
5240 RETURN
5250 '
5260      *PAUSE1
5270 '
5280 WHILE L-1000<>ZZZ%
5290 GOSUB *MON
5295 IF L-1000=ZZZ% THEN FOR J=0 TO CINT(256/AVN) :GOSUB *MON :NEXT J
```

```
5300 WEND
5310 RETURN
5320 '
5330      *PAUSE2
5340 '
5350 LOCATE 0,0 :PRINT SPACE$(32) :LOCATE 0,0
5360 PRINT "NOW PAUSING ! PRESS N OR A !" :COLOR@ (0,0)-(31,0),2 :BEEP
5370 C$="" :WHILE C$="" :C$=INKEY$ :WEND
5380 IF C$="n" OR C$="N" THEN AT=0
5390 IF C$="a" OR C$="A" THEN AT=1
5400 LOCATE 0,0 :PRINT SPACE$(32)
5410 RETURN
5420 '
5430      *SETCON1
5440 '
5450 LOCATE 0,1 :PRINT SPACE$(17) :LOCATE 0,1
5460 PRINT "SELECT(m)" :COLOR@ (0,1)-(10,1),4
5470 LOCATE 0,2 :PRINT SPACE$(20) :LOCATE 0,2
5480 PRINT "[0/4/5/6/*]" :COLOR@ (0,2)-(20,2),4
5490 M1=0 :WHILE M1=0 :GOSUB *MON :WEND
5500 RETURN
5510 '
5520      *SETCON2
5530 '
5540 LOCATE 0,1 :PRINT SPACE$(17) :LOCATE 0,1
5550 PRINT "SELECT(s)" :COLOR@ (0,1)-(10,1),4
5560 LOCATE 0,2 :PRINT SPACE$(20) :LOCATE 0,2
5570 PRINT "[0/d/o/a/w/r/v]" :COLOR@ (0,2)-(20,2),4
5580 B$="" :WHILE B$="" :B$=INKEY$ :WEND
5590 'IF B$="0" THEN GOSUB *ZERO
5600 IF B$="d" OR B$="D" THEN LOCATE 10,11 :INPUT "",DPR!
5610 IF B$="d" OR B$="D" THEN LOCATE 9,11 :PRINT SPACE$(10)
5620 IF B$="d" OR B$="D" THEN LOCATE 9,11 :PRINT DPR!
5630 IF B$="o" OR B$="O" THEN LOCATE 10,12 :INPUT "",OPR!
5640 IF B$="o" OR B$="O" THEN LOCATE 9,12 :PRINT SPACE$(10)
5650 IF B$="o" OR B$="O" THEN LOCATE 9,12 :PRINT OPR!
5660 IF B$="a" OR B$="A" THEN LOCATE 10,13 :INPUT "",AG!
5670 IF B$="a" OR B$="A" THEN LOCATE 9,13 :PRINT SPACE$(10)
5680 IF B$="a" OR B$="A" THEN LOCATE 9,13 :PRINT AG!
5690 IF B$="w" OR B$="W" THEN LOCATE 10,14 :INPUT "",WT
5700 IF B$="w" OR B$="W" THEN LOCATE 9,14 :PRINT SPACE$(10)
5710 IF B$="w" OR B$="W" THEN LOCATE 9,14 :PRINT WT
5720 IF B$="w" OR B$="W" THEN WT1!=WT :WT2!=WT :WT3!=WT
5730 IF B$="r" OR B$="R" THEN LOCATE 10,15 :INPUT "",RN
5740 IF B$="r" OR B$="R" THEN LOCATE 9,15 :PRINT SPACE$(10)
5750 IF B$="r" OR B$="R" THEN LOCATE 9,15 :PRINT RN
5760 IF B$="v" OR B$="V" THEN LOCATE 10,16 :INPUT "",DV!
5770 IF B$="v" OR B$="V" THEN LOCATE 9,16 :PRINT SPACE$(10)
```

```
5780 IF B$="v" OR B$="V" THEN LOCATE 9,16 :PRINT DV!
5790 RETURN
5800 '
5810      *SETCON3
5812 '
5814 LOCATE 0,1 :PRINT SPACE$(17) :LOCATE 0,1
5816 PRINT "SELECT(i)" :COLOR@ (0,1)-(10,1),4
5818 LOCATE 0,2 :PRINT SPACE$(20) :LOCATE 0,2
5820 PRINT "[2/4/6/8]" :COLOR@ (0,2)-(20,2),4
5822 B$="" :M1=0 :WHILE M1=0
5824 B$=INKEY$ :IVPO=0 :IVPW=0
5826 IF B$="2" THEN IVPW=-1
5828 IF B$="4" THEN IVPO=-1
5830 IF B$="6" THEN IVPO=1
5832 IF B$="8" THEN IVPW=1
5834 IF B$="2" OR B$="4" OR B$="6" OR B$="8" THEN GOSUB *DDPLY
5836 IF B$="2" OR B$="4" OR B$="6" OR B$="8" THEN GOSUB *IDPLY
5838 IF B$=CHR$(&HOD) THEN M1=1
5840 B$="" :WEND
5842 RETURN
5844 '
5846      *SETCON4
5848 '
5850 LOCATE 0,1 :PRINT SPACE$(17) :LOCATE 0,1
5852 PRINT "SELECT(t)" :COLOR@ (0,1)-(10,1),4
5854 LOCATE 0,2 :PRINT SPACE$(20) :LOCATE 0,2
5856 PRINT "[0/2/4/6/8/+]" :COLOR@ (0,2)-(20,2),4
5858 B$="" :M1=0 :WHILE M1=0
5860 B$=INKEY$ :TVPO=0 :TVPW=0
5862 IF B$="0" THEN TVS=-TVS
5864 IF B$="2" THEN TVPW=-1
5866 IF B$="4" THEN TVPO=-1
5868 IF B$="6" THEN TVPO=1
5870 IF B$="8" THEN TVPW=1
5871 IF B$="+" THEN GOSUB *PLASMA
5872 IF B$="0" OR B$="2" OR B$="4" OR B$="6" OR B$="8" THEN GOSUB *LDDPLY
5874 IF B$="0" OR B$="2" OR B$="4" OR B$="6" OR B$="8" THEN GOSUB *TDPLY
5875 IF B$="0" OR B$="2" OR B$="4" OR B$="6" OR B$="8" THEN GOSUB *EDPLY
5876 IF B$=CHR$(&HOD) THEN M1=1
5878 B$="" :WEND
5880 RETURN
5882 '
5884      *SETCON5
5886 '
5888 LOCATE 0,1 :PRINT SPACE$(17) :LOCATE 0,1
5890 PRINT "SELECT(e)" :COLOR@ (0,1)-(10,1),4
5892 LOCATE 0,2 :PRINT SPACE$(20) :LOCATE 0,2
5894 PRINT "[0/2/4/6/8/+]" :COLOR@ (0,2)-(20,2),4
```

```
5896 B$="" :M1=0 :WHILE M1=0
5898 B$=INKEY$ :EVP0=0 :EVPW=0
5900 IF B$="0" THEN EVS=-EVS
5902 IF B$="2" THEN EVPW=-1
5904 IF B$="4" THEN EVP0=-1
5906 IF B$="6" THEN EVP0=1
5908 IF B$="8" THEN EVPW=1
5909 IF B$="+" THEN GOSUB *PLASMA
5910 IF B$="0" OR B$="2" OR B$="4" OR B$="6" OR B$="8" THEN GOSUB *LDDPLY
5912 IF B$="0" OR B$="2" OR B$="4" OR B$="6" OR B$="8" THEN GOSUB *TDPLY
5913 IF B$="0" OR B$="2" OR B$="4" OR B$="6" OR B$="8" THEN GOSUB *EDPLY
5914 IF B$=CHR$(&H0D) THEN M1=1
5916 B$="" :WEND
5918 RETURN
7000 '
7010      *MON0
7020 '
7050 AVN=6                                    'IN%: Average num
7060 ZZ=-1 :PPP4!=0 :PPP5!=0                  'PPP4/5: Ex-counts on Z0#
7070 SS=0 :SSS=0 :M1=0 :M2=0 :ZPP=0 :ZPPS=0 :Z0#=0 :KS=0
7080 PA3=0 :PA4=0 :PA5=0 :PB3=0 :PB4=0 :PB5=0 :PPA4!=0 :PPA5!=0
7090 OUT &HD0,0
7100 '
7140 Z0#=LZ0*375/2 :ZPP=CINT(LZ0)
7150 LOCATE 0,4 :PRINT "[MONITOR]"
7160 LOCATE 0,5 :PRINT "Id(mA) "
7170 LOCATE 0,6 :PRINT "P(Torr)"
7180 LOCATE 0,7 :PRINT "Counter"
7190 LOCATE 0,8 :PRINT "Z(pt)  "
7200 '
7210      *MON
7220 '
7224 CH1=2 :CH2=3 :CH3=4 :CH4=5 :CH5=6        'Channel numbers of A/D
7226 CHS1=2^5 :CHS2=2^6
7230 DH1!=0 :DL1!=0 :DH2!=0 :DL2!=0 :DH3!=0 :DL3!=0
7235 DH4!=0 :DL4!=0 :DH5!=0 :DL5!=0
7240 '
7250 FOR I=1 TO AVN
7260 OUT &HD0,CH1+SS :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7265 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7270 DH1!=INP(&HD4)+DH1! :DL1!=INP(&HD6)+DL1!
7280 OUT &HD0,CH2+SS :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7285 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7290 DH2!=INP(&HD4)+DH2! :DL2!=INP(&HD6)+DL2!
7300 OUT &HD0,CH3+SS :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7305 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7310 DH3!=INP(&HD4)+DH3! :DL3!=INP(&HD6)+DL3!
7320 OUT &HD0,CH4+SS :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
```

```
7325 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7330 DH4!=INP(&HD4)+DH4! :DL4!=INP(&HD6)+DL4!
7340 OUT &HD0,CH5+SS :OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7345 OUT &HD2,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0 :OUT &H5F,0
7350 DH5!=INP(&HD4)+DH5! :DL5!=INP(&HD6)+DL5!
7360 GOSUB *MOVE1
7380 NEXT I
7390 '
7400 P1!=(256*DH1!+DL1!)/AVN :P2!=(256*DH2!+DL2!)/AVN
7410 P3!=(256*DH3!+DL3!)/AVN :P4!=(256*DH4!+DL4!)/AVN
7420 P5!=(256*DH5!+DL5!)/AVN
7430 GOSUB *COUNT
7440 GOSUB *MOVE1
7450 X!=(P1!-2048)*0.0025*10              'AMP=100
7460 Y!=(P2!-2048)*0.0025*2               'GI_REC=1/2
7470 Z!=Z0#*4/225 :ZP!=Z0#*2/375          'Z uni: mm / ZP uni: pt
7480 ZZZ=CINT(ZP!-0.5*ZPPS)
7500 IF ZZ<>-1 THEN GOSUB *MOVE2
7510 GOSUB *MOVE1
7520 LOCATE 9,5 :PRINT SPACE$(10) :LOCATE 9,5 :PRINT CINT(X!*10)/10
7530 LOCATE 9,6 :PRINT SPACE$(10) :LOCATE 9,6 :PRINT CINT(Y!*100)/100
7540 LOCATE 9,7 :PRINT SPACE$(10) :LOCATE 9,7 :PRINT Z0#
7550 LOCATE 9,8 :PRINT SPACE$(10) :LOCATE 9,8 :PRINT CINT(ZP!*10)/10
7560 IF Z!<-0.5 AND ZPPS=-1 THEN SS=0 :SSS=0 :ZPPS=0 :ZZ=-1 :M2=0
7570 IF Z!>450.5 AND ZPPS=1 THEN SS=0 :SSS=0 :ZPPS=0 :ZZ=-1 :M2=0
7590 RETURN
7600 '
7610 *MOVE1
7612 K$="" :K$=INKEY$
7614 IF K$="0" THEN LOCATE 10,8 :PRINT SPACE$(10)
7616 IF K$="0" THEN LOCATE 10,8 :INPUT "",ZP              'INP Z[pt]
7618 IF K$="0" THEN Z0#=ZP*375/2 :ZPP%=CINT(ZP)
7620 IF K$="0" THEN SS=0 :SSS=0 :ZPPS=0 :ZZ=-1           'RESET
7625 IF K$="5" THEN SS=0 :SSS=0
7630 IF K$="5" THEN ZPP=ZZZ :ZPPS=0 :ZZ=-1 :M2=0         'KEY5 & S* OFF
7635 IF K$="6" THEN SS=CHS1 :ZPPS=1 :ZZ=-1               'KEY6 & S* ON
7640 IF K$="4" THEN SS=CHS2 :ZPPS=-1 :ZZ=-1              'KEY4 & S* ON
7650 IF K$="*" THEN SS=0 :SSS=0 :ZPPS=0 :M2=0 :KS=1
7660 OUT &HD0,SS
7670 IF K$="*" THEN LOCATE 10,1 :PRINT SPACE$(5)
7680 IF K$="*" THEN LOCATE 10,1 :INPUT "",ZZ             'INPT Z[pt]
7690 IF K$=CHR$(&H0D) THEN M1=1
7700 RETURN
7710 '
7720 *MOVE2
7730 IF ZP!<ZZ AND M2=0 THEN SS=CHS1 :ZPPS=1 :M2=1          'S* ON
7740 IF ZP!>ZZ AND M2=0 THEN SS=CHS2 :ZPPS=-1 :M2=1         'S* ON
7750 IF ZP!=ZZ AND M2=0 THEN SS=0 :ZPPS=0 :M2=1
```

```
7760 ZZZ=CINT(ZP!-0.5*ZPPS)
7780 IF ZZZ=ZZ AND M2=1 THEN SS=0 :SSS=0                    'S* OFF
7790 IF ZZZ=ZZ AND M2=1 THEN ZPP=ZZZ :ZPPS=0 :ZZ=-1 :M2=0
7800 OUT &HD0,SS
7810 RETURN
7820 '
7830 *COUNT
7840 IF P3!>3072 THEN PA3=1 ELSE PA3=0
7850 IF P4!>3072 THEN PA4=1 :PP4=1 ELSE PA4=0 :PP4=0
7860 IF P5!>3072 THEN PA5=1 :PP5=1 ELSE PA5=0 :PP5=0
7870 IF PA3-PB3=-1 THEN PP3=1 ELSE PP3=0
7880 IF PA4-PB4=-1 THEN PPA4!=PPP4! ELSE PPA4!=0
7890 IF PA5-PB5=-1 THEN PPA5!=PPP5! ELSE PPA5!=0
7900 Z0#=Z0#+PP3*(PP4-PP5)+PPA4!-PPA5!
7910 PB3=PA3 :PB4=PA4 :PB5=PA5
7920 RETURN
```

```
#Initial File for LAN.EXE
#
DVR,       4.85
DPR,       2.5
OUT-R,     50
AMP-G,     300
WT-TIME,   10
RPT-NMB,   1
V-STEP,    0.2
V1,        1
V2,        5
V3,        9
VW,        2
SAVE-DIR, B:¥CALM2¥
```

```
/* d2u.c ********************************************************** Since Jul 2002 */
/*                                                                                  */
/*  Change DOS to Unix format of text files                                         */
/*                                                Programed by TAKEDA Tsuyoshi      */
/*                                                                                  */
/*      d2u [file-name]                                                             */
/*                                                                                  */
/*      e.g.) d2u 00000000.dat                                                      */
/*                                                                                  */
/*      NOTICE!  Not available for changing KANJI-codes.                            */
/*                                                                                  */
/************************************************** Last Modified in Jul 2002 */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


int rwfile(char []);

int main(int argc, char *argv[]){


  int  i;


  if (argc < 2){
    printf("¥nInput file-name !¥n");
    printf("d2u  [file-name]¥n¥n");
    return -1;
  }

  for(i=1; i<argc; i++){
    if(rwfile(argv[i])!=0)
      printf("%s not found !¥n",argv[i]);
  }


  return 0;


}

int rwfile(char *name){


  int  i, n;
  char word[10000][512];
  FILE *f;


  if((f=fopen(name, "r")) == NULL)
    return -1;
  for(i=0; i<10000; i++){
    if(feof(f)){
      n=i; break;
    }
    fgets(word[i], sizeof word[0], f);
    n=strlen(word[i]);
    word[i][n-2]='¥n';
```

```
    word[i][n-1]='¥0';
  }
  fclose(f);

  f=fopen(name, "w");
  for(i=0; i<=n; i++)
    fputs(word[i], f);
  fclose(f);


  return 0;


}



/* tz3d.c ************************************************** Since Jul 2002 */
/*                                                                          */
/*    Wave data to 3D in time and space                                     */
/*                                              Programed by TAKEDA Tsuyoshi*/
/*                                                                          */
/*                                                                          */
/*      tz3d  [switch] [file_tail1] ([file_tail2])                          */
/*      switch:  -c$     Column number $ red in files (default, $ = 1).     */
/*               -f$     First number $ of the head of file-name            */
/*                       (default, $ = 0).                                  */
/*               -s$     Step number $ of the head of file-name             */
/*                       (default, $ = 1).                                  */
/*               -e$     End number $ of the head of file-name              */
/*                       (default, until final existant number).            */
/*               -o$     Offset value $ of red column (-c$)                 */
/*                       ($:integer, default:$ = 0).                        */
/*      file_tail1/2:    Tail which of file-name is common with each file.  */
/*                                                                          */
/*      e.g.) tz3d  -f000 -o-512  0000.890 0020.890                         */
/*                                                                          */
/*                                                                          */
/*      NOTICE!  Only available for average data and no using wild-card.    */
/*               Data not modified by range and offset.                     */
/*                                                                          */
/*               Max. limited points   1024                                 */
/*               Max. limited files     256                                 */
/*                                                                          */
/****************************************************** Last Modified in Oct 2002 */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>


struct  inidata {

  int  ac, cm, ft, sp, ed, os, sg, t, z;
  char av[8][128], ftail[4][128];

};


int  initial(struct  inidata  *);
int  readfile(struct  inidata  *,  int  [][256],  int  [][256]);
int  writefile(struct  inidata  *,  int  [][256],  int  [][256]);



int main(int  argc, char  *argv[]){


  int  i,  j,  k;
  int  rd1[1024][256],  rd2[1024][256];

  struct  inidata  id;

  id.ac = argc;
  for(i = 0; i < argc; i++)
    strcpy(id.av[i], argv[i]);
  id.cm = 1;  id.ft = 0;  id.sp = 1;  id.ed = 9;
  id.os = 0;  id.sg = 1;  id.t = 1024; id.z = 256;


  if(initial(&id) != 0)
    return -1;
  /*
  printf("ac=%d cm=%d ft=%d sp=%d ed=%d sg=%d¥n",
        id.ac, id.cm, id.ft, id.sp, id.ed, id.sg);
  printf("ftail1=%s¥tftail2=%s¥n",id.ftail[1],id.ftail[2]);
  */
  if(readfile(&id,  rd1,  rd2) != 0)
    return -1;
  /*
  printf("z=%d¥tt=%d¥n",id.t,id.z);
  */
  writefile(&id,  rd1,  rd2);
```

```
  return 0;

}

int  initial(struct  inidata  *d){

  int  i,  j,  m,  n,  in, s1,  s2;

  j = 1;  m = 0;  n = 0;  s1 =1;  s2 =1;
  in = d->ac;

  for(i = 1; i < in; i++){

    if(d->av[i][0] == '-'){
      switch(d->av[i][1]){
      case 'c':
        strcpy(d->av[i], d->av[i]+2);
        d->cm = atoi(d->av[i]);
        break;
      case 'f':
        strcpy(d->av[i], d->av[i]+2);
        d->ft = atoi(d->av[i]);
        s1 = strlen(d->av[i]);
        break;
      case 's':
        strcpy(d->av[i], d->av[i]+2);
        d->sp = atoi(d->av[i]);
        break;
      case 'e':
        strcpy(d->av[i], d->av[i]+2);
        d->ed = atoi(d->av[i]);
        s2 = strlen(d->av[i]);
        m = 1; break;
      case 'o':
        strcpy(d->av[i], d->av[i]+2);
        d->os = atoi(d->av[i]);
        break;
      default:
        printf("¥n");
        printf("tz3d  [switch] [file_tail1] ([file_tail2])¥n");
        printf("switch:  -c$     Column number $ red in files (default, $ = 1).¥n");
        printf("         -f$     First number $ of the head of file-name¥n");
        printf("                 (default, $ = 0). ¥n");
```

```
      printf("         -s$     Step number $ of the head of file-name¥n");
      printf("                 (default, $ = 1). ¥n");
      printf("         -e$     End number $ of the head of file-name¥n");
      printf("                 (default, until final existant number). ¥n");
      printf("         -o$     Offset value $ of red column (-c$)¥n");
      printf("                 ($:integer, default:$ = 0).¥n");
      printf("file_tail1/2:    Tail which of data-file-name is common with each file.¥n");
      printf("¥n");
      printf("e.g.) tz3d  -f000 -o-512  0000.890 0020.890¥n");
      printf("¥n¥n");
      printf("NOTICE!  Only available for average data and no using wild-card.¥n");
      printf("         Data not modified by range.¥n");
      printf("¥n");
      printf("         Max. limited points   1024¥n");
      printf("         Max. limited files     256¥n");
      printf("¥n");
      n = 1;
    }
  }
  else{
    if(j>2){
      printf("¥nFile-name too many !¥n¥n");
      n = -1; break;
    }
    d->ftail[2][0] = '¥0';
    strcpy(d->ftail[j], d->av[i]);
    j++;
  }

}

if(j == 1 && n == 0){
  printf("¥nFile-name not appointed !¥n¥n");
  n = -1;
}
if(s1 <= s2)
  d->sg =s2;
else
  d->sg =s1;
if(m == 0){
  d->ed = (int)pow(10,d->sg)-1;
  if(d->sp < 0)
    d->ed = 0;
}


return n;
```

```
}


int  readfile(struct  inidata  *d,  int  dd1[][256],  int  dd2[][256]){


  int  i,  j,  k,  m,  n,  i0,  in,  di,  kn,  sn;
  long int  fp;
  char  name1[128],  name2[128],  fname[128],  word[128];
  FILE  *f;

  n = 0;  m = (int)pow(10,d->sg);
  i0 = d->ft;  di = d->sp;  in = d->ed;  kn = d->cm;
  sn = 1;
  if(d->sp < 0)
    sn = -1;
  strcpy(name2, d->ftail[1]);


  for(i = i0; i*sn <= in*sn; i += di){

    sprintf(name1, "%d", m+i);
    strcpy(fname, name1+1);
    strcat(fname, name2);

    if((f=fopen(fname, "r")) == NULL){
      if(i == i0){
        printf("¥n%s not found !¥n¥n", fname);
        n = -1;
      }
      break;
    }
    else{
      for(j = 0; j < 128; j++){
        fp = ftell(f);
        fgets(word, 128, f);
        if(word[0] != '#')
          break;
      }
      fseek(f, fp, SEEK_SET);
      for(j = 0; j < 1024; j++){
        for(k = 1; k <= kn; k++)
          fscanf(f, "%d", &dd1[j][i]);
        fgets(word, 128, f);
        if(feof(f))
          break;
```

```
    }
    fclose(f);
    if(d->t > j)
      d->t = j;
  }

}
if(d->z > i)
  d->z = i;

strcpy(name2, d->ftail[2]);
if(name2[0] != '¥0'){
for(i = i0; i*sn <= in*sn; i += di){

  sprintf(name1, "%d", m+i);
  strcpy(fname, name1+1);
  strcat(fname, name2);

  if((f=fopen(fname, "r")) == NULL){
    if(i == i0){
      printf("¥n%s not found !¥n¥n", fname);
      n = -1;
    }
    break;
  }
  else{
    for(j = 0; j < 128; j++){
      fp = ftell(f);
      fgets(word, 128, f);
      if(word[0] != '#')
        break;
    }
    fseek(f, fp, SEEK_SET);
    for(j = 0; j < 1024; j++){
      for(k = 1; k <= kn; k++)
        fscanf(f, "%d", &dd2[j][i]);
      fgets(word, 128, f);
      if(feof(f))
        break;
    }
    fclose(f);
    if(d->t > j)
      d->t = j;
  }

}
if(d->z > i)
  d->z = i;
```

```
  }

  return n;

}

int  writefile(struct  inidata  *d,  int  dd1[][256],  int  dd2[][256]){

  int  c,  dd0,  i,  in,  j,  j0,  jn,  dj,  sn;
  char  name[8],  name1[128],  name2[128],  fname[128];
  FILE  *f;

  c = d->cm;  dd0 = d->os;  in = d->t;
  j0 = d->ft;  dj = d->sp;  jn = d->z;
  sn = 1;
  if(d->sp < 0)
    sn = -1;

  sprintf(name, ".c%d", c);
  strcpy(name1, d->ftail[1]);
  strcpy(name2, d->ftail[2]);
  strcpy(fname, name1);
  if(name2[0] != '¥0'){
    strcat(fname, ".");
    strcat(fname, name2);
  }
  strcat(fname, name);
  strcat(fname, ".tz3d");

  f=fopen(fname, "w");
  fprintf(f, "#t   z  %s[%d]", name1, c);
  if(name2[0] != '¥0')
    fprintf(f, "  %s[%d]-%s[%d]", name1, c, name2, c);
  fprintf(f, "¥n");

  for(i = 0; i < in; i++){

    fprintf(f, "¥n");
    for(j = j0; j*sn < jn*sn; j += dj){
      fprintf(f, "%d¥t%d¥t%d", i, j, dd1[i][j]+dd0);
      if(name2[0] != '¥0')
        fprintf(f, "¥t%d", dd1[i][j]-dd2[i][j]);
```

```
      fprintf(f, "¥n");
    }

  }

  fclose(f);


  return 0;

}



/* zv3d.c ******************************************************* Since Aug 2002 */
/*                                                                                */
/*    Wave data to 3D in space and potential                                      */
/*                                                Programed by TAKEDA Tsuyoshi    */
/*                                                                                */
/*                                                                                */
/*      tz3d  [switch] [file_tail]                                                */
/*      switch:  -c$     Column number $ red in files (default, $ = 1).           */
/*               -fh$    First number $ of the head of file-name                  */
/*                       (default, $ = 0).                                        */
/*               -sh$    Step number $ of the head of file-name                   */
/*                       (default, $ = 1).                                        */
/*               -eh$    End number $ of the head of file-name                    */
/*                       (default, to no existant number).                        */
/*               -fc$    First number $ of the chest of file-name                 */
/*                       (default, $ = 0).                                        */
/*               -sc$    Step number $ of the chest of file-name                  */
/*                       (default, $ = 1).                                        */
/*               -ec$    End number $ of the chest of file-name                   */
/*                       (default, until final existant number).                  */
/*               -o$     Offset value $ of red column (-c$)                       */
/*                       ($:integer, default:$ = 0).                              */
/*      file_tail:       Tail which of file-name is common with each file.        */
/*                                                                                */
/*      e.g.) zv3d  -fh000 -fc000 -sc2 -o-512  0.890                              */
/*                                                                                */
/*                                                                                */
/*      NOTICE!  Only available for average data and no using wild-card.          */
/*               Data not modified by range and offset.                           */
/*                                                                                */
/*               Max. limited points   1024                                       */
/*               Max. limited files    256*128                                    */
/*                                                                                */
```

```
/******************************************************* Last Modified in Oct 2002 */



#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>


struct  inidata {

  int  ac, cm, t, z, v, os;
  int  fth, sph, edh, sgh, ftc, spc, edc, sgc;
  char av[8][128], ftail[128];

};


int  initial(struct  inidata  *);
int  readfile(struct  inidata  *,  int  [][256][128]);
int  writefile(struct  inidata  *,  int  [][256][128]);



int main(int  argc, char  *argv[]){


  int  i,  j,  k;
  static int  rd[1024][256][128];

  struct  inidata  id;

  id.ac = argc;
  for(i = 0; i < argc; i++)
    strcpy(id.av[i], argv[i]);
  id.cm = 1;  id.t = 1024;  id.z = 256;  id.v =128;  id.os = 0;
  id.fth = 0;  id.sph = 1;  id.edh = 9;  id.sgh = 1;
  id.ftc = 0;  id.spc = 1;  id.edc = 9;  id.sgc = 1;


  if(initial(&id) != 0)
    return -1;
  /*
  printf("ac=%d cm=%d¥n", id.ac, id.cm);
  printf("fth=%d sph=%d edh=%d sgh=%d¥n",
         id.fth, id.sph, id.edh, id.sgh);
  printf("ftc=%d spc=%d edc=%d sgc=%d¥n",
```

```
        id.ftc, id.spc, id.edc, id.sgc);
  printf("ftail=%s¥n",id.ftail);
  */
  if(readfile(&id,  rd) != 0)
    return -1;
  /*
  printf("t=%d¥tz=%d¥tv=%d¥n", id.t, id.z, id.v);
  */
  writefile(&id,  rd);


  return 0;


}


int  initial(struct  inidata  *d){


  int  i,  j,  n,  in;
  int  mh,  mc,  sh1,  sh2,  sc1,  sc2;

  j = 1;  mh = 0;  mc = 0;  n = 0;
  sh1 = 1;  sh2 = 1; sc1 = 1; sc2 = 1;
  in = d->ac;


  for(i = 1; i < in; i++){

    if(d->av[i][0] == '-'){
      switch(d->av[i][1]){
      case 'c':
        strcpy(d->av[i], d->av[i]+2);
        d->cm = atoi(d->av[i]);
        break;
      case 'f':
        if(d->av[i][2] == 'h'){
          strcpy(d->av[i], d->av[i]+3);
          d->fth = atoi(d->av[i]);
          sh1 = strlen(d->av[i]);
          break;
        }
        else{
          if(d->av[i][2] == 'c'){
            strcpy(d->av[i], d->av[i]+3);
            d->ftc = atoi(d->av[i]);
            sc1 = strlen(d->av[i]);
```

```
        break;
      }
    }
    n = 1; break;
  case 's':
    if(d->av[i][2] == 'h'){
      strcpy(d->av[i], d->av[i]+3);
      d->sph = atoi(d->av[i]);
      break;
    }
    else{
      if(d->av[i][2] == 'c'){
        strcpy(d->av[i], d->av[i]+3);
        d->spc = atoi(d->av[i]);
        break;
      }
    }
    n = 1; break;
  case 'e':
    if(d->av[i][2] == 'h'){
      strcpy(d->av[i], d->av[i]+3);
      d->edh = atoi(d->av[i]);
      sh2 = strlen(d->av[i]);
      mh = 1; break;
    }
    else{
      if(d->av[i][2] == 'c'){
        strcpy(d->av[i], d->av[i]+3);
        d->edc = atoi(d->av[i]);
        sc2 = strlen(d->av[i]);
        mc = 1; break;
      }
    }
    n = 1; break;
  case 'o':
    strcpy(d->av[i], d->av[i]+2);
    d->os = atoi(d->av[i]);
    break;
  default:
    n = 1;
  }
}
else{
  if(j>1){
    printf("¥nFile-name too many !¥n¥n");
    n = -1; break;
  }
  strcpy(d->ftail, d->av[i]);
```

```
    j++;
  }

}

if(n == 1){
  printf("¥n");
  printf("zv3d  [switch] [file_tail]¥n");
  printf("switch:  -c$     Column number $ red in files (default, $ = 1).¥n");
  printf("         -fh$    First number $ of the head of file-name¥n");
  printf("                 (default, $ = 0). ¥n");
  printf("         -sh$    Step number $ of the head of file-name¥n");
  printf("                 (default, $ = 1). ¥n");
  printf("         -eh$    End number $ of the head of file-name¥n");
  printf("                 (default, to no existant number). ¥n");
  printf("         -fc$    First number $ of the chest of file-name¥n");
  printf("                 (default, $ = 0). ¥n");
  printf("         -sc$    Step number $ of the chest of file-name¥n");
  printf("                 (default, $ = 1). ¥n");
  printf("         -ec$    End number $ of the chest of file-name¥n");
  printf("                 (default, until final existant number). ¥n");
  printf("         -o$     Offset value $ of red column (-c$)¥n");
  printf("                 ($:integer, default:$ = 0).¥n");
  printf("file_tail:       Tail which of data-file-name is common with each file.¥n");
  printf("¥n");
  printf("e.g.) zv3d  -fh000 -fc000 -sc2 -o-512  0.890¥n");
  printf("¥n¥n");
  printf("NOTICE!  Only available for average data and no using wild-card.¥n");
  printf("         Data not modified by range.¥n");
  printf("¥n");
  printf("         Max. limited points   1024¥n");
  printf("         Max. limited files    256*128¥n");
  printf("¥n");
}
else{
  if(j == 1){
    printf("¥nFile-name not appointed !¥n¥n");
    n = -1;
  }
  if(sh1 <= sh2)
    d->sgh = sh2;
  else
    d->sgh = sh1;
  if(mh == 0){
    d->edh = (int)pow(10,d->sgh)-1;
    if(d->sph < 0)
      d->edh = 0;
  }
```

```
    if(sc1 <= sc2)
      d->sgc = sc2;
    else
      d->sgc = sc1;
    if(mc == 0){
      d->edc = (int)pow(10,d->sgc)-1;
      if(d->spc < 0)
        d->edc = 0;
    }
  }


  return n;


}



int  readfile(struct  inidata  *d,  int  dd[][256][128]){


  int  h,  i,  h0,  hn,  dh,  i0,  in,  di;
  int  j,  k,  n,  mh,  mc,  kn,  snh,  snc;
  long int  fp;
  char  name1[128],  name2[128],  name3[128],  fname[128],  word[128];
  FILE  *f;

  n = 0;
  mh = (int)pow(10,d->sgh);
  mc = (int)pow(10,d->sgc);
  h0 = d->fth;  dh = d->sph;  hn = d->edh;
  i0 = d->ftc;  di = d->spc;  in = d->edc;
  kn = d->cm;
  snh = 1;  snc = 1;
  if(d->sph < 0)
    snh = -1;
  if(d->spc < 0)
    snc = -1;
  strcpy(name3, d->ftail);


  for(h = h0; h*snh <= hn*snh; h += dh){

    sprintf(name1, "%d", mh+h);

    for(i = i0; i*snc <= in*snc; i += di){
```

```
      sprintf(name2, "%d", mc+i);
      strcpy(fname, name1+1);
      strcat(fname, name2+1);
      strcat(fname, name3);

      if((f=fopen(fname, "r")) == NULL){
        if(h == h0 && i == i0){
          printf("¥n%s not found !¥n¥n", fname);
          n = -1;
        }
        break;
      }
      else{
        for(j = 0; j < 128; j++){
          fp = ftell(f);
          fgets(word, 128, f);
          if(word[0] != '#')
            break;
        }
        fseek(f, fp, SEEK_SET);
        for(j = 0; j < 1024; j++){
          for(k = 1; k <= kn; k++)
            fscanf(f, "%d", &dd[j][h][i]);
          fgets(word, 128, f);
          if(feof(f))
            break;
        }
        fclose(f);
        if(d->t > j)
          d->t = j;
      }

    }
    if(n == -1 || i == i0)
      break;
    if(d->v > i)
      d->v = i;

  }
  if(d->z > h)
    d->z = h;


  return n;


}
```

```
int  writefile(struct  inidata  *d,  int  dd[][256][128]){


  int  i,  j,  i0,  in,  di,  j0,  jn,  dj;
  int  c,  dd0,  h,  hn,  snh,  snc;
  char  name1[128],  name2[8],  name3[8],  fname[128];
  FILE  *f;

  c = d->cm;   dd0 = d->os;  hn = d->t;
  i0 = d->fth;  di = d->sph;  in = d->z;
  j0 = d->ftc;  dj = d->spc;  jn = d->v;
  snh = 1;  snc = 1;
  if(d->sph < 0)
    snh = -1;
  if(d->spc < 0)
    snc = -1;


  strcpy(name1, d->ftail);
  sprintf(name2, ".c%d.", c);

  for(h = 0; h < hn; h++){

    sprintf(name3, "%d", 10000+h);
    strcpy(fname, name1);
    strcat(fname, name2);
    strcat(fname, name3+1);
    strcat(fname, ".zv3d");

    f=fopen(fname, "w");
    fprintf(f, "#z   v  (z)(v)%s[%d]  (z)(v)%s[%d]-(z)(v+dv)%s[%d]¥n",
            name1, c, name1, c, name1, c);
    for(i = i0; i*snh < in*snh; i += di){
      fprintf(f, "¥n");
      for(j = j0; j*snc < (jn-dj)*snc; j += dj)
        fprintf(f, "%d¥t%d¥t%d¥t%d¥n", i, j, dd[h][i][j]+dd0, dd[h][i][j]-dd[h][i][j+dj]);
    }

    fclose(f);

  }


  return 0;
```

```
}


/* fft3d.c ********************************************************* Since Aug 2002 */
/*                                                                                 */
/*    FFT of 3D for time and space                                                 */
/*                                                 Programed by TAKEDA Tsuyoshi    */
/*                                                                                 */
/*                                                                                 */
/*      fft3d  [switch] [file]                                                     */
/*      switch:  -c$     Column number $ red in files (default:$ = 3).             */
/*               -fb$    First block number $ of column 1                          */
/*                       ($:0 - 1023, default:$ = 0).                              */
/*               -sb$    Step block number $ of column 1                           */
/*                       ($:1 - 1023, default:$ = 1).                              */
/*               -eb$    End block number $ of column 1                            */
/*                       ($:0 - 1023, default:until existant final number).        */
/*               -fp$    First points number $ of column 2                         */
/*                       ($:0 - 255, default:$ = 0).                               */
/*               -sp$    Step points number $ of column 2                          */
/*                       ($:1 - 255, default:$ = 1).                               */
/*               -ep$    End points number $ of column 2                           */
/*                       ($:0 - 255, default:until existant final number).         */
/*      file:            File-name which will be analized by fft.                  */
/*                                                                                 */
/*      e.g.) fft3d  -fb100 -eb355  *.tz3d                                         */
/*                                                                                 */
/*                                                                                 */
/*      NOTICE!  Only available for integer (average) data.                        */
/*               Max. limited points   1024*256                                    */
/*               Max. limited files    1024                                        */
/*                                                                                 */
/*********************************************************** Last Modified in Oct 2002 */



#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>


struct  inidata {

  int  ac, cm, f, h;
  int  ftp, spp, edp, ftb, spb, edb;
  char av[1040][128], file[1024][128];
```

```
};

struct  data {

  int  t,  z,  d[1024][256];
  double  wk[1024][256];

};


int  initial(struct inidata  *);
int  readfile(struct inidata  *,  struct  data  *,  int);
int  writefile(struct inidata  *,  struct data  *,  int);
int  spectrum(struct data  *);
int  fft(int,  double  [],  double  []);
/*int  FFT(int,  double  [],  double  []);*/

int main(int  argc,  char  *argv[]){


  int  i,  in,  j,k;
  int  rd[1024][256];
  static struct inidata  id;
  static struct data  d;

  id.ac = argc;
  for(i = 0; i < argc; i++)
    strcpy(id.av[i], argv[i]);
  id.cm = 3;  id.h = 1;
  id.ftb = 0;  id.spb = 1;  id.edb = 1023;
  id.ftp = 0;  id.spp = 1;  id.edp = 255;
  d.t = 1024; d.z = 256;

  if(initial(&id) != 0)
    return -1;
  /*
  printf("ac=%d cm=%d f=%d h=%d¥n", id.ac, id.cm, id.f, id.h);
  printf("ftb=%d spb=%d edb=%d¥n", id.ftb, id.spb, id.edb);
  printf("ftp=%d spp=%d edp=%d¥n", id.ftp, id.spp, id.edp);
  */

  in = id.f;
  for(i = 0; i < in; i++){

    if(readfile(&id, &d, i) != 0)
      return -1;
    /*
```

```
    printf("%d file=%s¥n", i, id.file[i]);
    printf("edb=%d  edp=%d t=%d z=%d¥n", id.edb, id.edp, d.t, d.z);
    */
    spectrum(&d);
    writefile(&id, &d, i);

  }


  return 0;


}


int  initial(struct inidata  *d){


  int  i,  j,  n,  in;
  float  hh;

  j = 0;  n = 0;
  in = d->ac;


  for(i = 1; i < in; i++){

    if(d->av[i][0] == '-'){
      switch(d->av[i][1]){
      case 'c':
        strcpy(d->av[i], d->av[i]+2);
        d->cm = atoi(d->av[i]);
        break;
      case 'f':
        if(d->av[i][2] == 'p'){
          strcpy(d->av[i], d->av[i]+3);
          d->ftp = atoi(d->av[i]);
          break;
        }
        else{
          if(d->av[i][2] == 'b'){
            strcpy(d->av[i], d->av[i]+3);
            d->ftb = atoi(d->av[i]);
            break;
          }
        }
        n = 1; break;
      case 's':
```

```
      if(d->av[i][2] == 'p'){
        strcpy(d->av[i], d->av[i]+3);
        d->spp = atoi(d->av[i]);
        break;
      }
      else{
        if(d->av[i][2] == 'b'){
          strcpy(d->av[i], d->av[i]+3);
          hh = atof(d->av[i]);
          if(hh == 0.5){
            d->h = 2;
            d->spb = 1;
          }
          else{
            d->h = 1;
            d->spb = atoi(d->av[i]);
          }
          break;
        }
      }
      n = 1; break;
    case 'e':
      if(d->av[i][2] == 'p'){
        strcpy(d->av[i], d->av[i]+3);
        d->edp = atoi(d->av[i]);
        break;
      }
      else{
        if(d->av[i][2] == 'b'){
          strcpy(d->av[i], d->av[i]+3);
          d->edb = atoi(d->av[i]);
          break;
        }
      }
      n = 1; break;
    default:
      n = 1;
    }
  }
  else{
    strcpy(d->file[j], d->av[i]);
    j++;
  }

}

d->f = j;
```

```
  if(n == 1){
    printf("¥n");
    printf("fft3d  [switch] [file]¥n");
    printf("switch:  -c$     Column number $ red in files (default:$ = 3).¥n");
    printf("         -fb$    First block number $ of column 1¥n");
    printf("                 ($:0 - 1023, default:$ = 0).¥n");
    printf("         -sb$    Step block number $ of column 1¥n");
    printf("                 ($:1 - 1023, default:$ = 1).¥n");
    printf("         -eb$    End block number $ of column 1¥n");
    printf("                 ($:0 - 1023, default:until existant final number).¥n");
    printf("         -fp$    First points number $ of column 2¥n");
    printf("                 ($:0 - 255, default:$ = 0).¥n");
    printf("         -sp$    Step points number $ of column 2¥n");
    printf("                 ($:1 - 255, default:$ = 1).¥n");
    printf("         -ep$    End points number $ of column 2¥n");
    printf("                 ($:0 - 255, default:until existant final number).¥n");
    printf("file:            File-name which will be analized by fft.¥n");
    printf("¥n");
    printf("e.g.) fft3d  -fb100 -eb355  *.tz3d¥n");
    printf("¥n¥n");
    printf("NOTICE!  Only available for integer (average) data.¥n");
    printf("         Max. limited points   1024*256¥n");
    printf("         Max. limited files    1024.¥n");
    printf("¥n");
  }
  else{
    if(j == 0){
      printf("¥nFile-name not appointed !¥n¥n");
      n = -1;
    }
    if(j>1024){
      printf("¥nFile too many !¥n¥n");
      n = -1;
    }
  }


  return n;


}


int  readfile(struct inidata  *d,  struct data  *dd,  int  fn){


  int  i,  j,  k,  k_,  m,  x,  y,  z,  snp,  snb;
```

```
int  i0,  di,  in,  j0,  dj,  jn,  dx,  zn;
long int  fp;
char  fname[128],  name[128];
static char word[1024][256][64];
FILE  *f;


strcpy(fname, d->file[fn]);
if((f=fopen(fname, "r")) == NULL){
  printf("¥n%s not found !¥n¥n", fname);
  return -1;
}
for(i = 0; i < 32; i++){
  fp = ftell(f);
  fgets(name, 64, f);
  if(name[0] != '#' && name[0] != '¥n')
    break;
}
fseek(f, fp, SEEK_SET);

m = 0;
for(i = 0; i < 1024; i++){

  for(j = 0; j <= 256; j++){
    fgets(word[i][j], 64, f);
    if(feof(f)){
      m = 1;  break;
    }
    if(word[i][j][0] == '¥n'){
      break;
    }
  }
  if(m != 0)
    break;
  if(d->edp > j && j > 0)
    d->edp = j-1;

}
fclose(f);

if(d->edb > i){
  if(m != 0 && j != 0)
    d->edb = i;
  else
    d->edb = i-1;
}

i0 = d->ftb;  di = d->spb;  in = d->edb;
```

```
j0 = d->ftp;  dj = d->spp;  jn = d->edp;
dx = d->h;   zn = d->cm;
snp = 1;  snb = 1;
if(d->spp < 0)
  snp = -1;
if(d->spb < 0)
  snb = -1;

x = 0;
for(i = i0; i*snb <= in*snb; i += di){
  y = 0;
  for(j = j0; j*snp <= jn*snp; j += dj){
    z = 0; k_ = 0;
    for(k = 0; k < 64; k++){
      if(z == zn)
        break;
      else{
        if(word[i][j][k] == ' ' || word[i][j][k] == '¥t' || word[i][j][k] == '¥n'){
          strcpy(name, word[i][j]+k_);
          name[k] = '¥0';
          dd->d[x][y] = atoi(name);
          if(dx == 2 && x > 0)
            dd->d[x-1][y] = (dd->d[x][y]+dd->d[x-2][y])/2;
          k_ = k; z++;
          if(word[i][j][k] == '¥n')
            break;
        }
      }
    }
    y++;
  }
  if(dd->z > y)
    dd->z = y;
  x += dx;
}
if(d->h==1 && dd->t > x)
  dd->t = x;
if(d->h==2)
  dd->t = x;

for(i = 1; i <= 8; i++){
  if(dd->z < pow(2,i)){
    dd->z = (int)pow(2,i-1);
    d->edp = j0 + dj*((int)pow(2,i-1)-1);
    break;
  }
}
```

```
  if(dd->t != 500 && dd->t != 1000){
    for(i = 1; i <= 10; i++){
      if(d->h==1 && dd->t < pow(2,i)){
        dd->t = (int)pow(2,i-1);
        d->edb = i0 + di*((int)pow(2,i-1)-1);
        break;
      }
      if(d->h==2 && dd->t < pow(2,i)){
        dd->t = (int)pow(2,i-1);
        d->edb = i0 + di*((int)pow(2,i-2)-1);
        break;
      }
    }
  }
  if(dd->t == 500){
    for(i = 500; i < 512; i++){
      for(j = 0; j < dd->z; j++){
        dd->d[i][j] = 0;
      }
    }
    dd->t = 512;
  }
  if(dd->t == 1000){
    for(i = 1000; i < 1024; i++){
      for(j = 0; j < dd->z; j++){
        dd->d[i][j] = 0;
      }
    }
    dd->t = 1024;
  }


  return 0;


}


int  writefile(struct inidata  *d,  struct data  *dd,  int  fn){


  int  c,  i,  j,  in,  jn;
  char  fname[128],  name[128];
  FILE  *f;

  c = d->cm;  in = dd->t;  jn = dd->z;
```

```
  strcpy(name, d->file[fn]);
  if(d->h ==1)
    sprintf(name, "%s.c%d.b%d_%d_%d.p%d_%d_%d", name, c, d->ftb, d->edb, d->spb, d->ftp,
d->edp, d->spp);
  if(d->h ==2)
    sprintf(name, "%s.c%d.b%d_%d_0.5.p%d_%d_%d", name, c, d->ftb, d->edb,  d->ftp, d->edp,
d->spp);

  strcpy(fname, name);
  strcat(fname, ".wk3d");
  f=fopen(fname, "w");
  fprintf(f, "#w/2pi*n*dt¥tk/2pi*n*dz¥tP(w,k)/dt/dz¥n");
  for(i = 0; i < in/2; i++){
    fprintf(f, "¥n");
    for(j = 0; j < jn/2; j++)
      fprintf(f, "%d¥t%d¥t%e¥n", i, j, dd->wk[i][j]);
  }
  fclose(f);

  /*
  strcpy(fname, name);
  strcat(fname, ".test");
  f=fopen(fname, "w");
  for(i = 0; i < in; i++){
    fprintf(f, "¥n");
    for(j = 0; j < jn; j++)
      fprintf(f, "%d¥t%d¥t%d¥n", i, j, dd->d[i][j]);
  }
  fclose(f);
  */

  return 0;


}


int spectrum(struct data  *dd){


  int  i,  j,  in,  jn,  m,  n;
  double  o, x, x_, pi;
  static double  d1r[1024][256],  d1i[1024][256];
  static double  d2r[256][1024],  d2i[256][1024];

  pi = M_PI;
  in = dd->t;  jn = dd->z;
```

```
/*
m = int(jn*0.05); n = int(in*0.05);
for(i = 0; i < in;  i++){
  for(j = 0; j < jn;  j++){
    x = 1; x_ = 1;
    if(j < m){
      o = pi*(j/(double)m-1);
      x = 0.5*cos(o)+0.5;
    }
    if(i < n){
      o = pi*(i/(double)n-1);
      x_ = 0.5*cos(o)+0.5;
    }
    if(j > jn-m-1){
      o = pi*(j-jn+m+1)/(double)m;
      x = 0.5*cos(o)+0.5;
    }
    if(i > in-n-1){
      o = pi*(i-in+n+1)/(double)n;
      x_ = 0.5*cos(o)+0.5;
    }
    dd->d[i][j] = int(x*x_*dd->d[i][j]);
  }
}
*/

for(j = 0; j < jn;  j++){
  for(i = 0; i < in;  i++){
    d2r[j][i] = dd->d[i][j];
    d2i[j][i] = 0;
  }
  fft(in,  d2r[j],  d2i[j]);
}

for(i = 0; i < in;  i++){
  for(j = 0; j < jn;  j++){
    d1r[i][j] = d2r[j][i];
    d1i[i][j] = 0;
  }
  fft(jn,  d1r[i],  d1i[i]);
  for(j = 0; j < jn;  j++)
    dd->wk[i][j] = (d1r[i][j]*d1r[i][j] + d1i[i][j]*d1i[i][j])/in/jn;
}


return 0;
```

```
}


int fft(int  m,  double  ar[],  double  ai[]){


  int  i, j, k, n, p, q, x[1024], x_[1024];
  double  a1r_, a1i_, a2r_, a2i_, ar_, ai_;
  double  o, pi, wr, wi;

  pi = M_PI;
  frexp(m,&n);
  n = n-1;
  p = 1; q = m; o = pi/(double)m;


  x_[0]=0;
  for(i = 1; i <= n; i++){
    p = p*2; q = q/2; o = o*2;
    for(j = 0; j < p/2; j++){
      x[2*j] = x_[j]/2;
      x[2*j+1] = (x_[j] + m)/2;
      x_[2*j] = x[2*j];
      x_[2*j+1] = x[2*j+1];
      for(k = 0; k < q; k++){
        a1r_ = ar[k+2*j*q];
        a1i_ = ai[k+2*j*q];
        a2r_ = ar[k+(2*j+1)*q];
        a2i_ = ai[k+(2*j+1)*q];
        ar_ = a1r_ - a2r_;
        ai_ = a1i_ - a2i_;
        wr = cos(o*(double)k);  wi = sin(o*(double)k);
        ar[k+2*j*q] = a1r_ + a2r_;
        ai[k+2*j*q] = a1i_ + a2i_;
        ar[k+(2*j+1)*q] = ar_*wr - ai_*wi;
        ai[k+(2*j+1)*q] = ai_*wr + ar_*wi;
      }
    }
  }
  for(i = 0; i < m; i++){
    if(i < x[i]){
      ar_ = ar[i];
      ai_ = ai[i];
      ar[i] = ar[x[i]];
      ai[i] = ai[x[i]];
      ar[x[i]] = ar_;
      ai[x[i]] = ai_;
    }
```

```
  }

  return 0;

}

/*
int FFT(int  n,  double  ar[],  double  ai[]){


  int  m, mh, i, j, k;
  double  wr, wi, xr, xi, theta;

  theta = 2*M_PI/n;

  i = 0;
  for(j = 1; j < n-1; j++){
    for(k = n >> 1; k > (i ^= k); k >>= 1);
      if(j < i){
        xr = ar[j];
        xi = ai[j];
        ar[j] = ar[i];
        ai[j] = ai[i];
        ar[i] = xr;
        ai[i] = xi;
      }
  }
  theta *= n;
  for(mh = 1; (m = mh << 1) <= n; mh = m){
    theta *= 0.5;
    for(i = 0; i < mh; i++){
      wr = cos(theta*i);
      wi = sin(theta*i);
      for(j = i; j < n; j += m){
        k = j + mh;
        xr = wr*ar[k] - wi*ai[k];
        xi = wr*ai[k] + wi*ar[k];
        ar[k] = ar[j] - xr;
        ai[k] = ai[j] - xi;
        ar[j] += xr;
        ai[j] += xi;
      }
    }
  }

  return 0;

}
```

```
*/


/* square-average.c ********************************************** Since Mar 2003 */
/*                                                                               */
/*    Square Average for 3D files (.zv3d etc)                                    */
/*                                                Programed by TAKEDA Tsuyoshi   */
/*                                                                               */
/*                                                                               */
/*      square-average  [switch] [file-name]                                     */
/*      switch:    -c$     Column number $ red in files (default:$ = 3).         */
/*                 -b$     Averaging block number $                              */
/*                         ($:0-511, default:$ = 0).                             */
/*                 -p$     Averaging point number $                              */
/*                         ($:0-511, default:$ = 0).                             */
/*                                                                               */
/*      e.g.) square-average -c4 -b1 -p1 0.890.c1.0000.zv3d                      */
/*                                                                               */
/*                                                                               */
/*      NOTICE!  Square averaged is (2b+1)x(2p+1).                               */
/*               Only available for integer (average) data.                      */
/*               Max. limited points   1024*256                                  */
/*               Max. limited files    1024                                      */
/*                                                                               */
/******************************************************** Last Modified in Apr 2003 */



#include <stdio.h>
#include <stdlib.h>
#include <string.h>


struct  inidata {

  int  ac, cm, bn, pn, fn;
  char av[1040][128], name[1024][128];

};

struct  data {

  int  b,  p,  d[1024][256];
  double  dd[1024][256];

};
```

```
int  initial(struct inidata  *);
int  readfile(struct inidata  *,  struct  data  *,  int);
int  s_average(struct inidata  *,  struct  data  *);
int  writefile(struct inidata  *,  struct data  *,  int);


int main(int  argc,  char  *argv[]){


  int  i,  j,  k;
  int  i0,  in,  jn,  kn;
  int  rd[1024][256];
  static struct inidata  id;
  static struct data  d;

  id.ac = argc;
  for(i = 0; i < argc; i++)
    strcpy(id.av[i], argv[i]);
  id.cm = 3;  id.fn = 1;
  d.b = 1024; d.p = 256;

  if(initial(&id) != 0)
    return -1;
  /*
  printf("¥nac=%d cm=%d bn=%d pn=%d fn=%d¥n", id.ac, id.cm, id.bn, id.pn, id.fn);
  */

  i0 = 0;  in = id.fn;
  for(i = i0; i < in; i++){

    if(readfile(&id, &d, i) != 0)
      return -1;
    s_average(&id, &d);
    writefile(&id, &d, i);


  }

  return 0;


}


int  initial(struct inidata  *d){
```

```
int  i,  j,  n,  in;

j = 0;  n = 0;  in = d->ac;

for(i = 1; i < in; i++){

  if(d->av[i][0] == '-'){
    switch(d->av[i][1]){
    case 'c':
      strcpy(d->av[i], d->av[i]+2);
      d->cm = atoi(d->av[i]);
      break;
    case 'b':
      strcpy(d->av[i], d->av[i]+2);
      d->bn = atoi(d->av[i]);
      break;
    case 'p':
      strcpy(d->av[i], d->av[i]+2);
      d->pn = atoi(d->av[i]);
      break;
    default:
      n = 1;
    }
  }
  else{
    strcpy(d->name[j], d->av[i]);
    j++;
  }

}

if(n == 1){
  printf("¥n");
  printf("square-average  [switch] [file-name]¥n");
  printf("switch:    -c$     Column number $ red in files (default:$ = 3).¥n");
  printf("           -b$     Averaging block number $ ¥n");
  printf("                   ($:0-511, default:$ = 0).¥n");
  printf("           -p$     Averaging point number $¥n");
  printf("                   ($:0-511, default:$ = 0).¥n");
  printf("¥n");
  printf("e.g.) square-average -c4 -b1 -p1 0.890.c1.0000.zv3d¥n");
  printf("¥n¥n");
  printf("NOTICE!  Square averaged is (2b+1)x(2p+1).¥n");
  printf("         Only available for integer (average) data.¥n");
  printf("         Max. limited points   1024*256¥n");
  printf("         Max. limited files    1024.¥n");
  printf("¥n");
}
```

```
  else{
    if(j == 0){
      printf("¥nFile-name not appointed !¥n¥n");
      n = -1;
    }
    else
      d->fn = j;
  }

  return n;


}



int  readfile(struct inidata  *d,  struct data  *dd,  int  ff){


  int  i,  j,  k,  k_,  m;
  int  in,  jn,  zn,  x,  y,  z;
  long int  fp;
  char  fname[128], name[128];
  static char word[1024][256][64];
  FILE  *f;


  strcpy(fname, d->name[ff]);
  if((f=fopen(fname, "r")) == NULL){
    printf("¥n%s not found !¥n¥n", fname);
    return -1;
  }
  for(i = 0; i < 32; i++){
    fp = ftell(f);
    fgets(name, 64, f);
    if(name[0] != '#' && name[0] != '¥n')
      break;
  }
  fseek(f, fp, SEEK_SET);

  m = 0;
  for(i = 0; i < 1024; i++){

    for(j = 0; j < 256; j++){
      fgets(word[i][j], 64, f);
      if(feof(f)){
        m = 1;  break;
      }
```

```
    if(word[i][j][0] == '¥n'){
      break;
    }
  }
  if(m != 0)
    break;
  if(dd->p > j && j > 0)
    dd->p = j;
}
fclose(f);

if(m != 0){
  if(j != 0)
    dd->b = i+1;
  else
    dd->b = i;
}
else
  dd->b = i+1;

in = dd->b; jn = dd->p;  zn = d->cm;

x = 0;
for(i = 0; i < in; i++){
  y = 0;
  for(j = 0; j < jn; j++){
    z = 0; k_ = 0;
    for(k = 0; k < 64; k++){
      if(z == zn)
        break;
      else{
        if(word[i][j][k] == ' ' || word[i][j][k] == '¥t' || word[i][j][k] == '¥n'){
          strcpy(name, word[i][j]+k_);
          name[k] = '¥0';
          dd->d[x][y] = atoi(name);
          k_ = k; z++;
          if(word[i][j][k] == '¥n')
            break;
        }
      }
    }
    y++;
  }
  if(dd->p > y)
    dd->p = y;
  x++;
}
if(dd->b > x)
```

```
    dd->b = x;

  return 0;

}

int  writefile(struct inidata  *d,  struct data  *dd,  int  ff){

  int  c,  i,  j,  in,  jn;
  char  fname[128], name[128];
  FILE  *f;

  in = dd->b;  jn = dd->p;

  strcpy(fname, d->name[ff]);
  sprintf(name, ".c%d.%d%dsav", d->cm, d->bn, d->pn);
  strcat(fname, name);
  f=fopen(fname, "w");
  fprintf(f, "#¥n");
  for(i = 0; i < in; i++){
    fprintf(f, "¥n");
    for(j = 0; j < jn; j++){
      fprintf(f, "%d¥t%d¥t%d¥n", i, j, (int)dd->dd[i][j]);
    }
  }
  fclose(f);

  return 0;

}

int  s_average(struct inidata  *d,  struct data  *dd){

  int  i,  j,  k,  m,  n;
  int  in,  jn,  mn,  nn;
  double  ddd;

  in = dd->b;  jn = dd->p;
  mn = d->bn;  nn = d->pn;
```

```
  for(i = 0; i < in; i++){
    for(j = 0; j < jn; j++){

      k = 0; ddd = 0;
      for(m = -mn; m <= mn; m++){
        for(n = -nn; n <= nn; n++){
          if(i+m >= 0 && j+n >= 0){
            ddd = ddd + dd->d[i+m][j+n];
            k++;
          }
        }
      }
      dd->dd[i][j] = ddd/k;

    }
  }


  return 0;


}



/* time-average.c ************************************************** Since Mar 2003 */
/*                                                                                 */
/*    Time-Average for 3D files (.zv3d etc)                                        */
/*                                                Programed by TAKEDA Tsuyoshi     */
/*                                                                                 */
/*                                                                                 */
/*      time-average  [switch] [head-name] ([tail-name])                           */
/*      switch:    -c$     Column number $ read in files (default:$ = 3).          */
/*                 -m$     Middle file number $.                                   */
/*                 -f$     Averaging file number $ ($:1-511, default:$ = 1).       */
/*      head-name:         Head of file-name.                                      */
/*      tail-name:         Tail of file-name (default: .zv3d).                     */
/*                                                                                 */
/*      e.g.) time-average -c4 -m0100 0.890.c2.                                    */
/*                                                                                 */
/*                                                                                 */
/*      NOTICE!  Not available for wild-card.                                      */
/*               Only available for integer (average) data.                        */
/*               Max. limited points   1024*256                                    */
/*               Max. limited files    1024                                        */
/*                                                                                 */
/********************************************************** Last Modified in Apr 2003 */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>


struct  inidata {

  int  ac, cm, mf, fn, dn;
  char av[8][128], name[5][128];

};

struct  data {

  int  b,  p,  d[1024][256];
  double  dd[1024][256];

};


int  initial(struct inidata  *);
int  readfile(char  [],  struct  data  *,  int);
int  writefile(char  [],  struct  data  *,  int);



int main(int  argc,  char  *argv[]){


  int  i,  j,  k,  os;
  int  i0,  in,  jn,  kn;
  int  rd[1024][256];
  char  fname[128], name0[128], name[128];
  struct inidata  id;
  struct data  d;

  id.ac = argc;
  for(i = 0; i < argc; i++)
    strcpy(id.av[i], argv[i]);
  id.cm = 3;  id.fn = 1;  id.dn = 1;
  strcpy(id.name[1],".zv3d");
  d.b = 1024; d.p = 256;

  if(initial(&id) != 0)
    return -1;
```

```
  /*
  printf("¥nac=%d cm=%d mf=%d fn=%d dn=%d¥n", id.ac, id.cm, id.mf, id.fn, id.dn);
  */

  i0 = id.mf - id.fn;
  in = id.mf + id.fn;
  os = (int)pow(10,id.dn);
  sprintf(name0, "%d", os+id.mf);

  for(i = i0; i <= in; i++){

    strcpy(fname, id.name[0]);
    sprintf(name, "%d", os+i);
    strcat(fname, name+1);
    strcat(fname, id.name[1]);

    if(readfile(fname, &d, id.cm) != 0)
      return -1;

    jn = d.b;  kn = d.p;
    for(j = 0; j < jn; j++){
      for(k = 0; k < kn; k++){
        if(i == i0)
          d.dd[j][k] = (double)d.d[j][k];
        else
          d.dd[j][k] = d.dd[j][k] + (double)d.d[j][k];
      }
    }

  }


  strcpy(fname, id.name[0]);
  strcat(fname, name0+1);
  strcat(fname, id.name[1]);
  sprintf(name, ".c%d.%dtav", id.cm, id.fn);
  strcat(fname, name);
  writefile(fname, &d, in-i0+1);


  return 0;


}


int  initial(struct inidata  *d){
```

```
int  i,  j,  n,  in;

j = 0;  n = 0;
in = d->ac;


for(i = 1; i < in; i++){

  if(d->av[i][0] == '-'){
    switch(d->av[i][1]){
    case 'c':
      strcpy(d->av[i], d->av[i]+2);
      d->cm = atoi(d->av[i]);
      break;
    case 'm':
      strcpy(d->av[i], d->av[i]+2);
      d->mf = atoi(d->av[i]);
      d->dn = strlen(d->av[i]);
      break;
    case 'f':
      strcpy(d->av[i], d->av[i]+2);
      d->fn = atoi(d->av[i]);
      break;
    default:
      n = 1;
    }
  }
  else{
    strcpy(d->name[j], d->av[i]);
    j++;
  }

}

if(n == 1){
  printf("¥n");
  printf("time-average  [switch] [head-name] ([tail-name])¥n");
  printf("switch:    -c$     Column number $ red in files (default:$ = 3).¥n");
  printf("           -m$     Middle file number $.¥n");
  printf("           -f$     Averaging file number $ ($:1-511, default:$ = 1).¥n");
  printf("head-name:         Head of file-name.¥n");
  printf("tail-name:         Tail of file-name (default: .zv3d).¥n");
  printf("¥n");
  printf("e.g.) time-average -c4 -m0100 0.890.c2.¥n");
  printf("¥n¥n");
  printf("NOTICE!  Not available for wild-card.¥n");
  printf("         Only available for integer (average) data.¥n");
```

```
    printf("         Max. limited points   1024*256¥n");
    printf("         Max. limited files    1024.¥n");
    printf("¥n");
  }
  else{
    if(j == 0){
      printf("¥nHead-name not appointed !¥n¥n");
      n = -1;
    }
    if(j>2){
      printf("¥nName too many !¥n¥n");
      n = -1;
    }
  }


  return n;


}



int  readfile(char  fname[],  struct data  *dd,  int  zn){


  int  i,  j,  k,  k_,  m;
  int  in,  jn,  x,  y,  z;
  long int  fp;
  char  name[128];
  static char word[1024][256][64];
  FILE  *f;


  if((f=fopen(fname, "r")) == NULL){
    printf("¥n%s not found !¥n¥n", fname);
    return -1;
  }
  for(i = 0; i < 32; i++){
    fp = ftell(f);
    fgets(name, 64, f);
    if(name[0] != '#' && name[0] != '¥n')
      break;
  }
  fseek(f, fp, SEEK_SET);

  m = 0;
  for(i = 0; i < 1024; i++){
```

```
  for(j = 0; j < 256; j++){
    fgets(word[i][j], 64, f);
    if(feof(f)){
      m = 1;  break;
    }
    if(word[i][j][0] == '¥n'){
      break;
    }
  }
  if(m != 0)
    break;
  if(dd->p > j && j > 0)
    dd->p = j;
}
fclose(f);

if(m != 0){
  if(j != 0)
    dd->b = i+1;
  else
    dd->b = i;
}
else
  dd->b = i+1;

in = dd->b; jn = dd->p;

x = 0;
for(i = 0; i < in; i++){
  y = 0;
  for(j = 0; j < jn; j++){
    z = 0; k_ = 0;
    for(k = 0; k < 64; k++){
      if(z == zn)
        break;
      else{
        if(word[i][j][k] == ' ' || word[i][j][k] == '¥t' || word[i][j][k] == '¥n'){
          strcpy(name, word[i][j]+k_);
          name[k] = '¥0';
          dd->d[x][y] = atoi(name);
          k_ = k; z++;
          if(word[i][j][k] == '¥n')
            break;
        }
      }
    }
    y++;
```

```
    }
    if(dd->p > y)
      dd->p = y;
    x++;
  }
  if(dd->b > x)
    dd->b = x;


  return 0;


}


int  writefile(char  fname[],  struct data  *dd,  int  fn){


  int  c,  i,  j,  in,  jn;
  char  name[128];
  FILE  *f;

  in = dd->b;  jn = dd->p;

  f=fopen(fname, "w");
  fprintf(f, "#¥n");
  for(i = 0; i < in; i++){
    fprintf(f, "¥n");
    for(j = 0; j < jn; j++){
      fprintf(f, "%d¥t%d¥t%d¥n", i, j, int(dd->dd[i][j]/fn));
    }
  }
  fclose(f);


  return 0;


}



/************************************************************** August 2002 **/
/**                                                                          **/
/**    Caluculation of Dispersion Ralation                                   **/
/**                                                                          **/
/**                                        Programed by TAKEDA Tsuyoshi      **/
/**                                                                          **/
```

```
/*****************************************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>

//double vr;
//double rr;
double nr;                /* Density Ratio  */
double_complex Dispersion(double_complex, double_complex);    /* Dispersion Relation */
void  DKA(double_complex  (*)(double_complex,double_complex),double_complex,double_complex
[],int,int,double,int *);


int main(){

  char name[80];
  char readf[20]="parameter.ini";
  char writef[20]="dispersion.txt";
  FILE *rf,*wf;

/* Definition of Variable */
  int i,j,m,n,eva;
  double val[20],esp,loop;
  double kr0,krn,dkr,ki0,kin,dki;
  double wr,wi,kr,ki,x,y,PI;
  double_complex a[10],k,w[10],w0[10];

/* Reading Initial Values */
  rf=fopen(readf,"rt");
  if(rf==NULL){
    printf("Not find '%s' !",readf);
    exit(1);
  }
  fgets(name,80,rf);
  for(i=1;i<=20;++i){
    fscanf(rf,"%s%lf",name,&val[i]);
    if(feof(rf))
      break;
  }
  fclose(rf);
  m=int(val[1]);esp=val[3];
  kr0=val[4];krn=val[5];dkr=val[6];
  ki0=val[7];kin=val[8];dki=val[9];
  nr=val[10];
```

```
/* Caluculation */
  j = 1;
  loop = (krn-kr0)/dkr+1;

  n = 4;
  PI = 3.141592653589793;
  k = double_complex(kr0,ki0);
  a[0] = -k*k;
  a[1] = k*double_complex(2,0);
  a[2] = k*k-double_complex(nr+1,0);
  a[3] = -k*double_complex(2,0);
  x = 0;
  for(i=0;i<=n-2;++i){
    y = real(a[i])*real(a[i])+imag(a[i])*imag(a[i]);
    if(y!=0)
      y = pow(y,1/(n-i)/2);
      if(y>x)
        x = y;
  }
  x = x*n;
  for(i=1;i<=n;++i)
    w0[i]                                                                                    =
-a[n-1]/double_complex(n,0)+double_complex(x,0)*exp(double_complex(0,1)*double_complex(2*P
I*(i-1)/n+PI/2/n,0));

  wf=fopen(writef,"wt");
  fprintf(wf,"#nr=%e¥n",nr);
  fprintf(wf,"#kr¥t¥tki¥t¥tw1r¥t¥tw1i¥t¥tw2r¥t¥tw2i¥t¥tw3r¥t¥tw3i¥t¥tw4r¥t¥tw4i¥n");
  for(kr=kr0;kr<=krn;kr+=dkr){
    for(i=1;i<=4;++i)
      w[i] = w0[i];
    for(ki=ki0;ki<=kin;ki+=dki){
      fprintf(wf,"%e¥t%e",kr,ki);
      k = double_complex(kr,ki);
      DKA(Dispersion,k,w,n,m,esp,&eva);
      if(eva==0){
        printf("¥nNot Converge in DKA !");
        exit(1);
          }
      for(i=1;i<=4;++i){
            wr = real(w[i]);
            wi = imag(w[i]);
        fprintf(wf,"¥t%e¥t%e",wr,wi);
      }
      fprintf(wf,"¥n");
      if(ki==ki0){
        for(i=1;i<=4;++i)
          w0[i] = w[i];
```

```
      }
    }
    printf("%2.0f%¥r",j/loop*100);
    ++j;
  }
  fclose(wf);

}


double_complex Dispersion(double_complex K,double_complex W){

/* Dispersion Relation */
  return
pow(W,4)-double_complex(2,0)*K*pow(W,3)+(K*K-double_complex(nr+1,0))*W*W+double_complex(2,
0)*K*W-K*K;

}


void        DKA(double_complex        (*func)(double_complex,double_complex),double_complex
x,double_complex y[],int n,int m,double esp,int *eva){

/* Definition of Variable */
  int j,k,l;
  double e;
  double_complex Multi,u[20],v[20];

  for(j=1;j<=m;++j){
    for(k=1;k<=n;++k){
      Multi = double_complex(1,1);
      for(l=1;l<=n;++l){
        if(l!=k)
          Multi = Multi*(y[k]-y[l]);
        u[k] = func(x,y[k])/Multi;
        v[k] = y[k]-u[k];
      }
    }
    e = 0;
    for(k=1;k<=n;++k){
      y[k] = v[k];
      e = e+pow(pow(real(u[k]/y[k]),2)+pow(imag(u[k]/y[k]),2),0.5);
    }
    if(e<esp){
      *eva = 1;
      break;
    }
    else
```

```
      *eva = 0;
  }

}



//Initial Data
m        1000
h        1e-12
esp      1e-8
kr0      0.0000000001
krn      2.0000000001
dkr      0.001
ki0      0.0000000001
kin      0.0000000001
dki      0.001
nr       0.012
```